Review: Deep Learning in Electron Microscopy

Jeffrey M. Ede[1,*]

[1]University of Warwick, Department of Physics, Coventry, CV4 7AL, UK
[*]j.m.ede@warwick.ac.uk

ABSTRACT

Deep learning is transforming most areas of science and technology, including electron microscopy. This review paper offers a practical perspective aimed at developers with limited familiarity. For context, we review popular applications of deep learning in electron microscopy. Following, we discuss hardware and software needed to get started with deep learning and interface with electron microscopes. We then review neural network components, popular architectures, and their optimization. Finally, we discuss future directions of deep learning in electron microscopy.

1 Introduction

Following decades of exponential increases in computational capability[1] and widespread data availability[2], scientists can routinely train artificial neural networks[3–10] (ANNs) to enable new science and technology[11–15]. The resulting deep learning revolution[16,17] has enabled superhuman performance in image classification[18–21], games[22–27], medical analysis[28,29], relational reasoning[30], speech recognition[31,32] and many other applications[33,34]. This introduction focuses on deep learning in electron microscopy and is aimed at developers with limited familiarity. For context, we therefore review popular applications of deep learning in electron microscopy. We then review resources available to support researchers and outline electron microscopy. Finally, we review popular ANN architectures and their optimization, and discuss future trends in artificial intelligence (AI) for electron microscopy.

Deep learning is motivated by universal approximator theorems[35–43], which state that sufficiently deep and wide[35,38,44] ANNs can approximate functions to arbitrary accuracy. It follows that ANNs can always match or surpass the performance of methods crafted by humans. In practice, DNNs reliably[45–47] learn generalizable[48–55] models without a prior understanding of physics. As a result, deep learning is freeing physicists from a need to devise equations to model complicated phenomena[12,13,15,56,57]. Most modern ANNs have millions of parameters, so inference often takes tens of milliseconds on GPUs or other hardware accelerators[58]. It is therefore unusual to develop ANNs to approximate computationally efficient methods with exact solutions, such as the fast Fourier transform[59–61] (FFT). However, ANNs are able to leverage an understanding of physics to accelerate time-consuming or iterative calculations[62–65], improve accuracy of methods[28,29,66], and find solutions that are otherwise intractable[22,67].

1.1 Improving Signal-to-Noise

A popular application of deep learning is to improve signal-to-noise[70,71]. For example, of medical electrical[72,73], medical image[74–76], optical microscopy[77–80], and speech[81–84] signals. There are many traditional denoising algorithms that are not based on deep learning[85–87], including linear[88,89] and non-linear[90–98] spatial domain filters, Weiner[99–101] filters, non-linear[102–107] wavelet domain filters, curvelet transforms[108,109], contourlet transforms[110,111], hybrid algorithms[112–118] that operate in both spatial and transformed domains, and dictionary-based learning[119–123]. However, traditional denoising algorithms are limited by features (often laboriously) crafted by humans and cannot exploit domain-specific context. In perspective, they leverage an ever-increasingly accurate representation of physics to denoise signals. However, traditional algorithms are limited by the difficulty of programmatically describing a complicated reality. Case in point, an ANN was able to outperform decades of advances in traditional denoising algorithms after training on two GPUs for a week[66].

Definitions of electron microscope noise can include statistical noise[124–126,126–131], aberrations[132], scan distortions[133–136], specimen drift[137], and electron beam damage[138]. Statistical noise is often minimized by traditional denoising algorithms[139,140], including a variety of denoising algorithms developed for electron microscopy. Examples include algorithms based on block matching[141], contourlet transforms[110,111], energy minimization[142], fast patch reorderings[143], Gaussian kernel density estimation[144], Kronecker envelope principal component analysis[145] (PCA), non-local means and Zernike moments[146], singular value thresholding[147], wavelets[148], and other approaches[137,149–152]. Noise that is not statistical is often minimized by hardware. For example, by using aberration correctors[132,153–155], choosing scan shapes and speeds that minimize distortions[134], and using stable sample holders to reduce drift[156]. Beam damage can also be reduced by using minimal electron voltage and electron

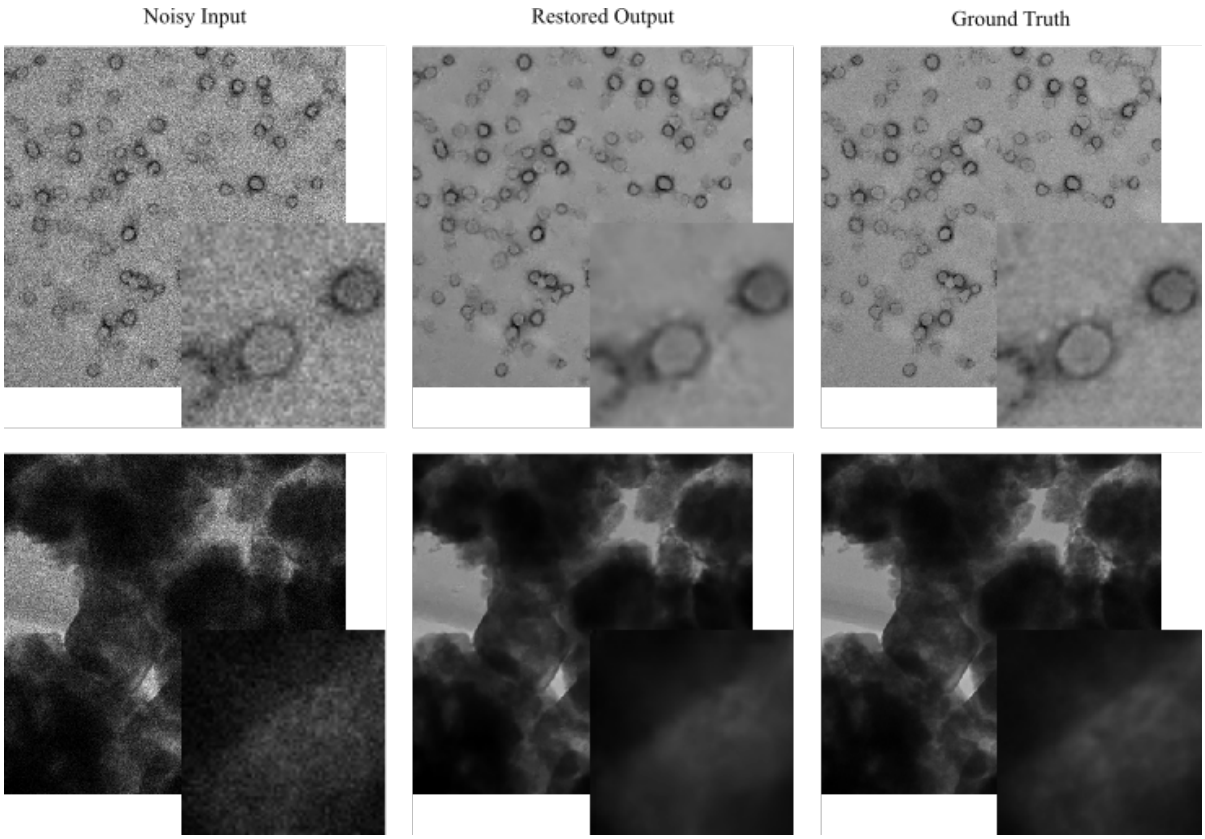

Figure 1. Example applications of a noise-removal DNN to instances of Poisson noise applied to 512×512 crops from TEM images. Enlarged 64×64 regions from the top left of each crop are shown to ease comparison. This figure is adapted from our earlier work[68] under a Creative Commons 4.0[69] license.

dose[157–159], or dose-fractionation across multiple frames in multi-pass TEM[160–162] or STEM[163].

Deep learning is being applied to improve signal-to-noise for a variety of applications[164–172]. Most approaches in electron microscopy involve training ANNs to either map low-quality experimental[173], artificially deteriorated[66,174] or synthetic[175–177] inputs to paired high-quality experimental measurements. For example, applications of a DNN trained with artificially deteriorated TEM images are shown in fig. 1. However, ANNs have also been trained with unpaired datasets of low-quality and high-quality electron micrographs[178], or pairs of low-quality electron micrographs[179,180]. Another approach is Noise2Void[164], ANNs are trained from single noisy images. However, Noise2Void removes information by masking noisy input pixels corresponding to target output pixels. So far, most ANNs that improve electron microscope signal-to-noise have been trained to decrease statistical noise[66,173,175,176,176–179,181]. Nevertheless, ANNs have been developed for aberration correction of optical microscopy[182–187] and photoacoustic[188] signals, and to correct electron microscope scan distortions[189,190] and specimen drift[137,190,191].

1.2 Compressed Sensing

Compressed sensing[194–198] is the reconstruction of a signal from a subset of measurements. Applications include image compression[199,200], faster medical imaging[201–203], lower medical radiation exposure[204–206], and low-light vision[207,208]. In electron microscopy, compressed sensing can enable electron beam exposure and scan time to be decreased by 10-100× with minimal information loss[192,193]. Thus, compressed sensing can be essential to investigations where the high current density of electron probes damages specimens[157,209–215]. Even if the effects of beam damage can be corrected by postprocessing, the damage to specimens is often permanent. Examples of beam-sensitive materials include organic crystals[216], metal-organic frameworks[217], nanotubes[218], and nanoparticle dispersions[219]. In electron microscopy, compressed sensing is especially effective due to high signal redundancy[220]. For example, most electron microscopy images are sampled at 5-10× their Nyquist rates[221] to ease visual inspection, decrease sub-Nyquist aliasing[222], and avoid undersampling.

Perhaps the most popular approach to compressed sensing is upsampling or infilling a uniformly spaced grid of signals[223–225]. Interpolation methods include Lancsoz[223], nearest neighbour[226], polynomial interpolation[227], Wiener[228] and other[229–231]

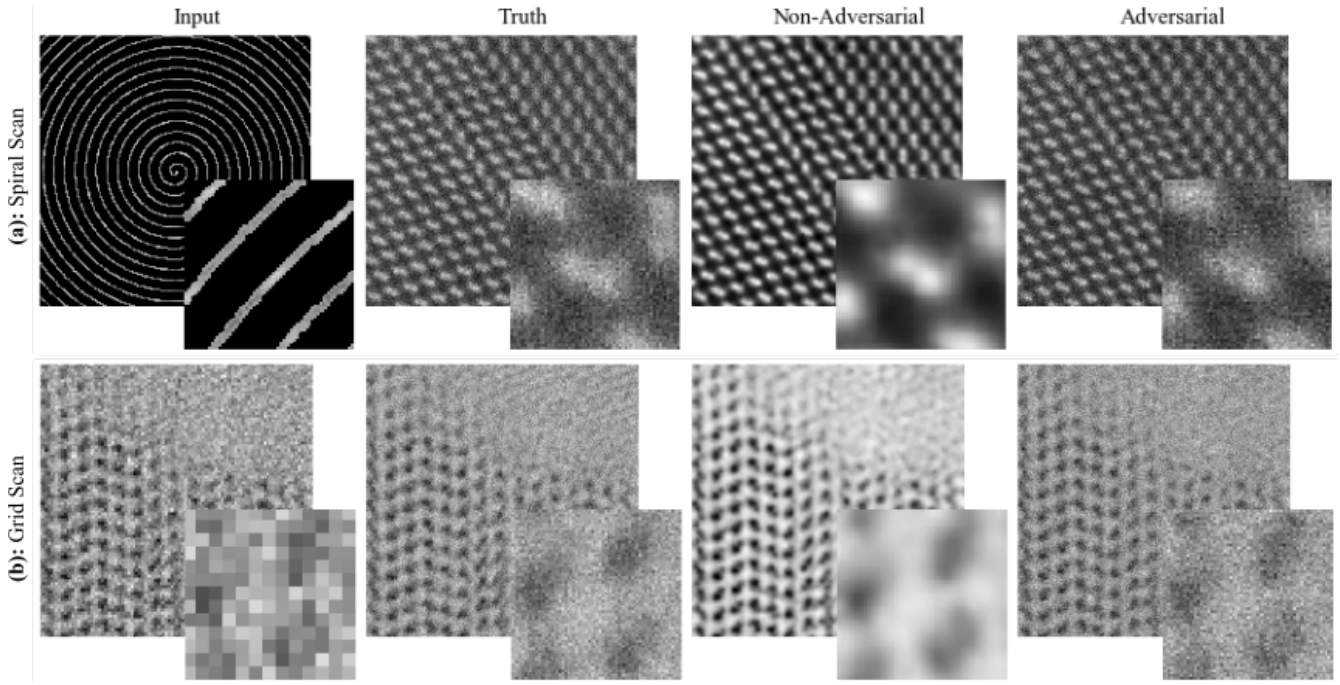

Figure 2. Example applications of DNNs to restore 512×512 STEM images from sparse signals. Adversarial training as part of a GAN yields more realistic outputs than training a single DNN with mean squared errors. Enlarged 64×64 regions from the top left of each crop are shown to ease comparison. a) Input is a Gaussian blurred 1/20 coverage spiral[192]. b) Input is a 1/25 coverage grid[193]. This figure is adapted from our earlier work under Creative Commons 4.0[69] licenses.

resampling. However, a variety of other strategies to minimize beam damage have also been proposed, including dose fractionation[232] and a variety of sparse data collection methods[233]. Perhaps the most intensively investigated approach to the latter is sampling a random subset of pixels, followed by reconstruction using an inpainting algorithm[233–238]. Random sampling of pixels is nearly optimal for reconstruction by compressed sensing algorithms[239]. However, random sampling exceeds the design parameters of standard electron beam deflection systems, and can only be performed by collecting data slowly[240,241], or with the addition of a fast deflection or blanking system[236,242].

Sparse data collection methods that are more compatible with conventional beam deflection systems have also been investigated. For example, maintaining a linear fast scan deflection whilst using a widely-spaced slow scan axis with some small random 'jitter'[234,240]. However, even small jumps in electron beam position can lead to a significant difference between nominal and actual beam positions in a fast scan. Such jumps can be avoided by driving functions with continuous derivatives, such as those for spiral and Lissajous scan paths[192,236,241,243,244]. Sang[241,244] considered a variety of scans including Archimedes and Fermat spirals, and scans with constant angular or linear displacements, by driving electron beam deflectors with a field-programmable gate array[245] (FPGA) based system. Spirals with constant angular velocity place the least demand on electron beam deflectors. However, dwell times, and therefore electron dose, decreases with radius. Conversely, spirals created with constant spatial speeds are prone to systematic image distortions due to lags in deflector responses. In practice, fixed doses are preferable as they simplify visual inspection and limit the dose dependence of STEM noise[125].

Deep learning can leverage an understanding of physics to infill images[246–248]. Example applications include increasing SEM[174,249,250], STEM[193,251] and TEM[252] resolution, and infilling continuous sparse scans[192]. Example applications of DNNs to complete sparse spiral and grid scans are shown in fig. 2. However, caution should be used when infilling large regions as ANNs may generate artefacts if a signal is unpredictable[192]. A popular alternative to deep learning for infilling large regions is exemplar-based infilling[253–256]. However, exemplar-based infilling often leaves artefacts[257] and is usually limited to leveraging information from single images. Smaller regions are often infilled by fast marching[258], Navier-Stokes infilling[259], or interpolation[227].

1.3 Labelling

Deep learning has been the basis of state-of-the-art classification[260–263] since convolutional neural networks (CNNs) enabled a breakthrough in classification accuracy on ImageNet[67]. Most classifiers are single feedforward neural networks (FNNs) that learn to predict discrete labels. In electron microscopy, applications include classifying potential imaging region quality[264],

material structures[265,266], and image quality[267]. However, siamese[268–270] and dynamically parameterized[271] networks can more quickly learn to recognise images. Finally, labelling ANNs can learn to predict continuous features, such as mechanical properties[272]. Labelling ANNs are often combined with other methods. For example, ANNs can be used to automatically identify particle locations[181,273–275] to ease subsequent processing.

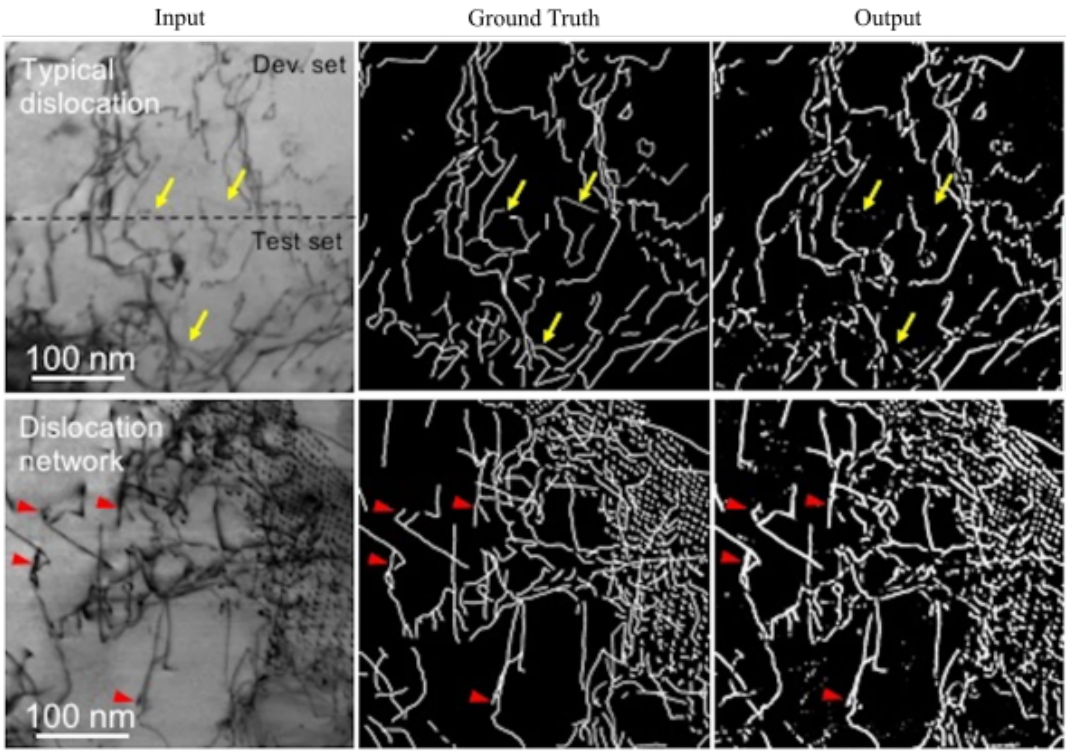

Figure 3. Example applications of a semantic segmentation DNN to STEM images of steel to predict dislocation locations. Yellow arrows mark uncommon dislocation lines with weak contrast, and red arrows indicate that fixed widths used for dislocation lines are sometimes too narrow to cover defects. This figure is adapted with permission[276] under a Creative Commons 4.0[69] license.

1.4 Semantic Segmentation

Semantic segmentation is the classification of pixels into discrete categories. In electron microscopy, applications include the automatic identification of local features[277,278], such as defects[279,280], dopants[281], material phases[282], material structures[283,284], dynamic surface phenomena[285], chemical phases in nanoparticles[286]. Early approaches to semantic segmentation used simple rules. However, such methods were not robust to a high variety of data[287]. Subsequently, more adaptive algorithms based on soft-computing[288] and fuzzy algorithms[289] were developed to use geometric shapes as priors. However, these methods were limited by programmed features and struggled to handle the high variety of data.

To improve performance, DNNs have been trained to semantically segment images[290–297]. Semantic segmentation DNNs have been developed for focused ion beam scanning electron microscopy[298–300] (FIB-SEM), SEM[300–303], STEM[276,304], and TEM[275,299,300,305–307]. For example, applications of a DNN to semantic segmentation of STEM images of steel are shown in fig. 3. Deep learning based semantic segmentation also has a high variety of applications outside of electron microscopy, including autonomous driving[308–312], dietary monitoring[313,314], magnetic resonance images[315–319], medical images[320–322] such as prenatal ultrasound[323–326], and translating satellite images[327–331]. Most DNNs for semantic segmentation are trained with images segmented by humans. However, human labelling may be too expensive, time-consuming, or inappropriate for sensitive data. Unsupervised semantic segmentation can avoid these difficulties by learning to segment images from an additional dataset

of segmented images[332] or image-level labels[333–336]. However, unsupervised semantic segmentation networks are often less accurate than supervised networks.

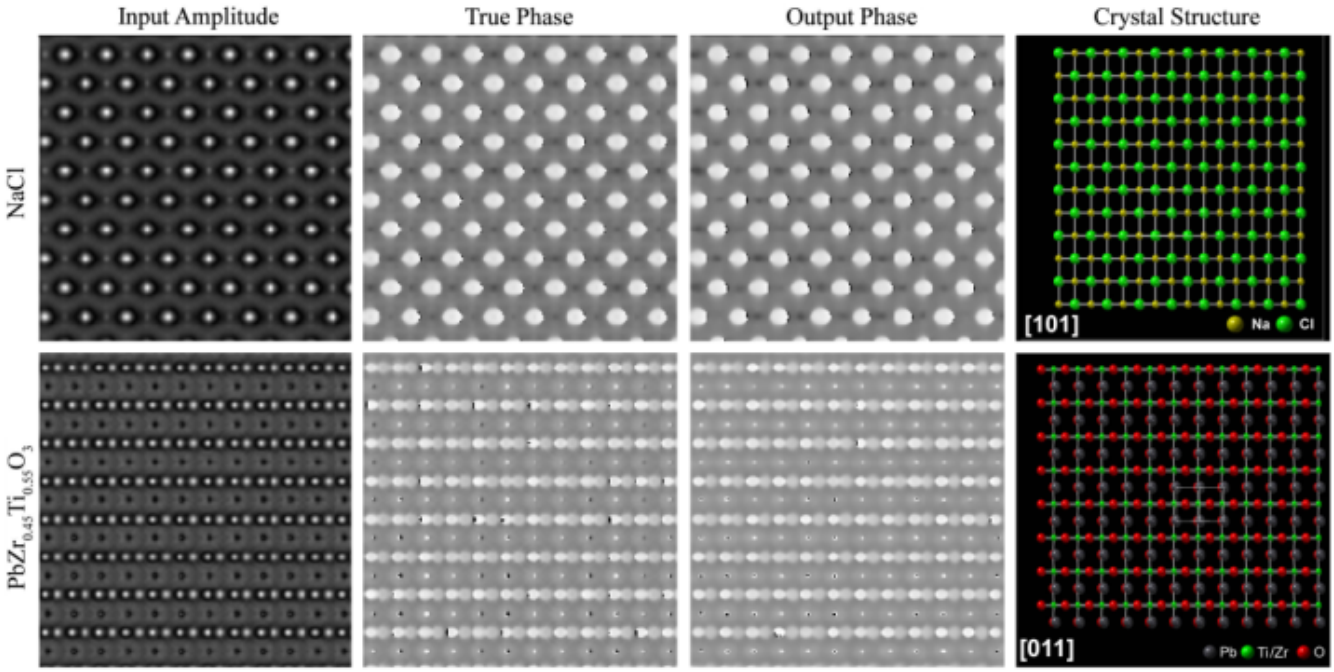

Figure 4. Example applications of a DNN to reconstruct phases of exit wavefunction from intensities of single TEM images. Phases in $[-\pi, \pi)$ rad are depicted on a linear greyscale from black to white, and Miller indices label projection directions. This figure is adapted from our earlier work[337] under a Creative Commons 4.0[69] license.

1.5 Exit Wavefunction Reconstruction

Electrons exhibit wave-particle duality[338,339], so electron propagation is often described by wave optics[340]. Applications of electron wavefunctions exiting materials[341] include determining projected potentials and corresponding crystal structure information[342,343], information storage, point spread function deconvolution, improving contrast, aberration correction[344], thickness measurement[345], and electric and magnetic structure determination[346,347]. Usually, exit wavefunctions are either iteratively reconstructed from focal series[348–352] or recorded by electron holography[340,351,353]. However, iterative reconstruction is often too slow for live applications, and holography is sensitive to distortions and may require expensive microscope modification.

Non-iterative methods based on DNNs have been developed to reconstruct optical exit wavefunctions from focal series[65] or single images[354–356]. Following, DNNs have been developed to reconstruct exit wavefunctions from single TEM images[337], as shown in fig. 4. Indeed, deep learning is increasingly being applied to accelerated quantum mechanics[357–362]. Other examples of DNNs adding new dimensions to data include semantic segmentation described in section 1.4, and reconstructing 3D atomic distortions from 2D images[363]. Non-iterative methods that do not use ANNs to recover phase information from single images have also been developed[364,365]. However, they are limited to defocused images in the Fresnel regime[364], or to non-planar incident wavefunctions in the Fraunhofer regime[365].

2 Resources

Access to scientific resources is essential to scientific enterprise[366]. Fortunately, most resources needed to get started with machine learning are freely available. This section provides directions to various machine learning resources, including how to access deep learning frameworks, a free GPU or TPU to accelerate tensor computations, platforms that host datasets and source code, and pretrained models. In support of Plan S[366–368], we focus on resources that enhance collaboration and enable open access[369]. We also discuss how electron microscopes can interface with ANNs and the importance of machine learning resources in the context of electron microscopy. However, we expect that our insights into electron microscopy can be generalized to other scientific fields.

2.1 Hardware Acceleration

Deep neural networks (DNNs) perform sequences of tensor operations. Tensors can either be computed on central processing units (CPUs) or hardware accelerators[58], such as field programmable gate arrays[370–373] (FPGAs), graphical processing units[374–376] (GPUs), and tensor processing units[377–379] (TPUs). Most benchmarks show that GPUs and TPUs outperform CPUs for typical DNNs that could be used for image processing[380–384] in electron microscopy. However, GPU and CPU performance can be comparable when CPU computation is optimized[385]. TPUs often outperform GPUs[382], and FPGAs can outperform GPUs[386,387] if FPGAs have sufficient arithmetic units[388,389]. Typical power consumption per TFLOPS[390] decreases in order CPU, GPU, FPGA, then TPU, so hardware acceleration can help to minimize long-term costs and environmental damage[391].

For beginners, Google Colab[392–395] and Kaggle[396] provide hardware accelerators in ready-to-go deep learning environments. Free compute time on these platforms is limited as they are not intended for industrial applications. Nevertheless, the free compute time is sufficient for some research[397]. For more intensive applications, it may be necessary to get permanent access to hardware accelerators. If so, many online guides detail how to install[398,399] and set up an Nvidia[400] or AMD[401] GPU in a desktop computer for deep learning. However, most hardware comparisons for deep learning[402] focus on Nvidia GPUs as most deep learning frameworks use Nvidia's proprietary cuDNN primitives for deep learning[403], which are optimized for Nvidia GPUs. Alternatively, hardware accelerators may be accessible from a university or other institutional high performance computing (HPC) centre, or via a public cloud service provider[404–407].

Framework	License	Programming Interfaces
Apache SINGA[408]	Apache 2.0[409]	C++, Java, Python
BigDL[410]	Apache 2.0[411]	Python, Scala
Caffe[412,413]	BSD[414]	C++, MATLAB, Python
Chainer[415]	MIT[416]	Python
Deeplearning4j[417]	Apache 2.0[418]	Clojure, Java, Kotlin, Python, Scala
Dlib[419,420]	BSL[421]	C++
Flux[422]	MIT[423]	Julia
MATLAB Deep Learning Toolbox[424]	Proprietary[425]	MATLAB
Microsoft Cognitive Toolkit[426]	MIT[427]	BrainScript, C++, Python
Apache MXNet[428]	Apache 2.0[429]	C++, Clojure, Go, JavaScript, Julia, Matlab, Perl, Python, R, Scala
OpenNN[430]	GNU LGPL[431]	C++
PaddlePaddle[432]	Apache 2.0[433]	C++
PyTorch[434]	BSD[435]	C++, Python
TensorFlow[436,437]	Apache 2.0[438]	C++, C#, Go, Haskell, Julia, MATLAB, Python, Java, JavaScript, R, Ruby, Rust, Scala, Swift
Theano[439,440]	BSD[441]	Python
Torch[442]	BSD[443]	C, Lua
Wolfram Mathematica[444]	Proprietary[445]	Wolfram Language

Table 1. Deep learning frameworks with programming interfaces. Most frameworks have open source code and many support multiple programming languages.

2.2 Deep Learning Frameworks

A deep learning framework[8,446–452] (DLF) is an interface, library or tool for DNN development. Features often include automatic differentiation[453], heterogeneous computation, pretrained models, and efficient computing[454] with CUDA[455–457], cuDNN[403,458] and OpenMP[459,460]. Popular DLFs tabulated in table 1 often have open source code and support multiple programming interfaces. Overall, TensorFlow[436,437] is the most popular DLF[461]. However, PyTorch[434] is the most popular DLF at top machine learning conferences[461,462]. Some DLFs also have extensions that ease development or extend functionality. For example, TensorFlow extensions[463] that ease development include Keras[464], Sonnet[465], Tensor2Tensor[466] and TFLearn[467,468], and extensions that add functionality include Addons[469], Agents[470], Dopamine[471], Federated[472–474], Probability[475] and TRFL[476]. In addition, DLFs are supplemented by libraries for predictive data analysis, such as scikit-learn[477].

A limitation of the DLFs in table 1 is that users must use programming interfaces. This is problematic as many electron microscopists have limited, if any, programming experience. To increase accessibility, a range of graphical user interfaces (GUIs) have been created for ANN development. For example, ANNdotNET[478], Create ML[479], Deep Cognition[480], Deep Network Designer[481], DIGITS[482], ENNUI[483], Expresso[484], Neural Designer[485], Waikato Environment for Knowledge Analysis[486–488] (WEKA) and ZeroCostDL4Mic[489]. The GUIs offer less functionality and scope for customization than programming interfaces. However, GUI-based DLFs are rapidly improving. Moreover, existing GUI functionality is more than sufficient to implement popular FNNs, such as image classifiers[262] and encoder-decoders[294–297,490–492].

2.3 Pretrained Models

Training ANNs is often time-consuming and computationally expensive[391]. Fortunately, pretrained models are available from a range of open access collections[493], such as Model Zoo[494], Open Neural Network Exchange[495–498] (ONNX) Model Zoo[499], TensorFlow Hub[500,501], and TensorFlow Model Garden[502]. Some researchers also provide pretrained models via project repositories[66,192,193,220,337]. Pretrained models can be used immediately or to transfer learning[503–509] to new applications. For example, by fine-tuning and augmenting the final layer of a pretrained model[510]. Benefits of transfer learning can include decreasing training time by orders of magnitude, reducing training data requirements, and improving generalization[508,511].

Using pretrained models is complicated by ANNs being developed with a variety of DLFs in a range of programming languages. However, most DLFs support interoperability. For example, by supporting the saving of models to a common format or to formats that are interoperable with the Neural Network Exchange Format[512] (NNEF) or ONNX formats. Many DLFs also support saving models to HDF5[513,514], which is popular in the pycroscopy[515,516] and HyperSpy[517,518] libraries used by electron microscopists. The main limitation of interoperability is that different DLFs may not support the same functionality. For example, Dlib[419,420] does not support recurrent neural networks[519–524] (RNNs).

2.4 Datasets

Randomly initialized ANNs[525] must be trained, validated, and tested with large, carefully partitioned datasets to ensure that they are robust to general use[526]. Most ANN training starts from random initialization, rather than transfer learning[503–509], as

1. Researchers may be investigating modifications to ANN architecture or ability to learn.
2. Pretrained models may be unavailable or too difficult to find.
3. Models may quickly achieve sufficient performance from random initialization. For example, training an encoder-decoder based on Xception[527] to improve electron micrograph signal-to-noise[66] can require less training than for PASCAL VOC 2012[528] semantic segmentation[294].
4. There may be a high computing budget, so transfer learning is unnecessary[529,530].

There are millions of open access datasets[531,532] and a range of platforms that host[533–537] or aggregate[538–541] machine learning datasets. Openly archiving datasets drives scientific enterprise by reducing need to repeat experiments[542–546], enabling new applications through data mining[547,548], and standardizing performance benchmarks[549]. For example, popular datasets used to standardize image classification performance benchmarks include CIFAR-10[550,551], MNIST[552] and ImageNet[553]. A high range of both domain-specific and general platforms that host scientific data for free are listed by the Open Access Directory[554] and Nature Scientific Data[555]. For beginners, we recommend Zenodo[556] as it is free, open access, has an easy-to-use interface, and will host an unlimited number of datasets smaller than 50 GB for at least 20 years[557].

There are a range of platforms dedicated to hosting electron microscopy datasets, including the Caltech Electron Tomography Database[558] (ETDB-Caltech), Electron Microscopy Data Bank[559–564] (EMDataBank), and the Electron Microscopy Public Image Archive[565,566] (EMPIAR). However, most electron microscopy datasets are small, esoteric or are not partitioned for machine learning[220]. Nevertheless, a variety of large machine learning datasets for electron microscopy are being published in independent repositories[220,567,568]. In addition, a variety of databases host information that supports electron microscopy. For example, crystal structure databases provide data in standard formats[569,570], such as Crystallography Information Files[571–574] (CIFs). Large crystal structure databases[575–577] containing over 10^5 crystal structures include the Crystallography Open Database[578–583] (COD), Inorganic Crystal Structure Database[584–588] (ICSD), and National Institute of Standards and Technology (NIST) Crystal Data[589,590].

To achieve high performance, it may be necessary to curate a large dataset. However, large datasets like DeepMind Kinetics[591], ImageNet[553], and YouTube 8M[592] may take a team months to prepare. As a result, it may not be practical to divert sufficient staff and resources to curate a high-quality dataset, even if curation is partially automated[592–599]. To curate data, human capital can be temporarily and cheaply increased by using microjob services[600]. For example, through microjob platforms tabulated in table 2. Increasingly, platforms are emerging that specialize in data preparation for machine learning. Nevertheless, microjob services may be inappropriate for sensitive data or tasks that require substantial domain-specific knowledge.

2.5 Source Code

Software is part of our cultural, industrial, and scientific heritage[601]. Source code should therefore be archived where possible. For example, on an open source code platform such as Apache Allura[602], AWS CodeCommit[603], Beanstalk[604], BitBucket[605], GitHub[606], GitLab[607], Gogs[608], Google Cloud Source Repositories[609], Launchpad[610], Phabricator[611], Savannah[612] or SourceForge[613]. These platforms enhance collaboration with functionality that helps users to watch[614] and contribute

Platform	Website	For Machine Learning
Amazon Mechanical Turk	https://www.mturk.com	General tasks
Appen	https://appen.com	Machine learning data preparation
Clickworker	https://www.clickworker.com	Machine learning data preparation
Fiverr	https://www.fiverr.com	General tasks
Hive	https://thehive.ai	Machine learning data preparation
iMerit	https://imerit.net	Machine learning data preparation
JobBoy	https://www.jobboy.com	General tasks
Minijobz	https://minijobz.com	General tasks
Microworkers	https://www.microworkers.com	General tasks
OneSpace	https://freelance.onespace.com	General tasks
Playment	https://playment.io	Machine learning data preparation
RapidWorkers	https://rapidworkers.com	General tasks
Scale	https://scale.com	Machine learning data preparation
Smart Crowd	https://thesmartcrowd.lionbridge.com	General tasks
Trainingset.ai	https://www.trainingset.ai	Machine learning data preparation
ySense	https://www.ysense.com	General tasks

Table 2. Microjob service platforms. The size of typical tasks varies for different platforms and some platforms specialize in preparing machine learning datasets.

improvements[615–621] to source code. The choice of platform is often not immediately important for small electron microscopy projects as most platforms offer similar functionality. Nevertheless, functionality comparisons of open source platforms are available[622–624]. For beginners, we recommend GitHub as it is actively developed, scalable to large projects and has an easy-to-use interface.

2.6 Finding Information

Most web traffic[625,626] goes to large-scale web search engines[627–631] such as Bing, DuckDuckGo, Google, Yahoo, and YouTube. This includes searches for scholarly content[632–634]. We recommend Google for electron microscopy queries as it appears to yield the best results for general[635–637], scholarly[633,634] and other[638] queries. However, general search engines can be outperformed by dedicated search engines for specialized applications. For example, for finding academic literature[639–641], data[642], jobs[643,644], publication venues[645], patents[646–649], people[650–652], and many other resources. The use of search engines is increasingly political[653–655] as they influence which information people see. However, most users appear to be satisfied with their performance[656].

Introductory textbooks are outdated[657,658] insofar that most information is readily available online. We find that some websites are frequent references for up-to-date and practical information:

1. Stack Overflow[659–664] is a source of working code snippets and a useful reference when debugging code.

2. Papers With Code State-of-the-Art[549] leaderboards rank the highest performing ANNs with open source code for various benchmarks.

3. Medium[665] and its subsidiaries publish blogs with up-to-date and practical advice about machine learning.

4. The Machine Learning subreddit[666] hosts discussions about machine learning. In addition, there is a Learn Machine Learning subreddit[667] aimed at beginners.

5. Dave Mitchell's DigitalMicrograph Scripting Website[668,669] hosts a collection of scripts and documentation for programming electron microscopes.

6. The Internet Archive[670,671] maintains copies of software and media, including webpages via its Wayback Machine[672–674].

7. Distill[675] is a journal dedicated to providing clear explanations about machine learning. Monetary prizes are awarded for excellent communication and refinement of ideas.

This list enumerates popular resources that we find useful, so it may introduce personal bias. However, alternative guides to useful resources are available[676–678]. We find that the most common issues finding information are part of an ongoing reproducibility crisis[679,680] where machine learning researchers are unwilling to publish their source code or data. Nevertheless, third party source code is sometimes available on GitHub. Alternatively, ANNs can reconstruct code from some research papers[681].

2.7 Scientific Publishing

The number of articles published per year in reputable peer-reviewed[682] scientific journals[683,684] has roughly doubled every nine years since the beginning of modern science[685]. There are now over 25000 peer-reviewed journals[684] with varying impact factors[686–688], scopes and editorial policies. Strategies to find the best journal to publish in include using online journal finders[689], seeking the advice of learned colleagues, and considering where similar research has been published. Increasingly, working papers are also being published in open access preprint archives before peer-review[690–692]. For example, the arXiv[693] is a popular preprint archive for computer science, mathematics, and physics. Advantages of preprints include ensuring that research is openly available, increasing discovery and citations[694,695], inviting timely scientific discussion, and raising awareness to reduce unnecessary duplication of research. Many publishers have adapted to the popularity of preprints[690] by offering open access publication options[696–699] and allowing, and in some cases encouraging[700], the prior publication of preprints. Indeed, some journals are now using the arXiv to host their publications[701].

A variety of software can help authors prepare scientific manuscripts[702]. However, we think the most essential software is a document preparation system. Most manuscripts are prepared with Microsoft Word[703] or similar software[704]. However, Latex[705–707] is a popular alternative among computer scientists, mathematicians and physicists[708]. Most electron microscopists at the University of Warwick appear to prefer Word. A 2014 comparison of Latex and Word found that Word is better at all tasks other than typesetting equations[709]. However, in 2017 it become possible to use Latex to typeset equations within Word[708]. As a result, Word appears to be more efficient than Latex for most manuscript preparation. Nevertheless, Latex may still be preferable to authors who want fine control over typesetting[710,711]. As a compromise, we use Overleaf[712] to edit Latex source code, then copy our code to Word as part of proofreading to identify issues with grammar and wording.

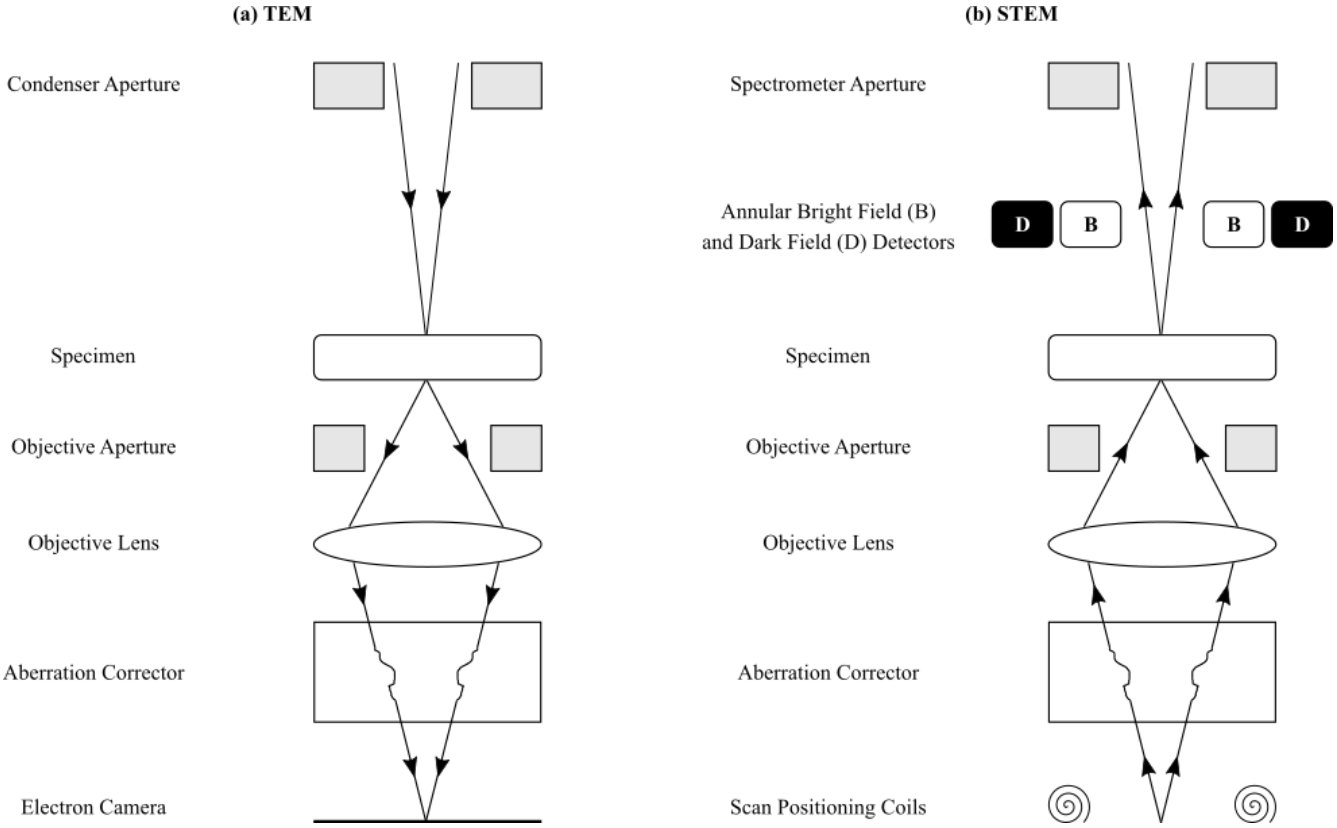

Figure 5. Reciprocity of TEM and STEM electron optics.

3 Electron Microscopy

An electron microscope is an instrument that uses electrons as a source of illumination to enable the study of small objects. Electron microscopy competes with a large range of alternative techniques for material analysis[713–715], including atomic force microscopy[716–718] (AFM); Fourier transformed infrared (FTIR) spectroscopy[719,720]; nuclear magnetic resonance[721–724] (NMR); Raman spectroscopy[725–731]; and x-ray diffraction[732,733] (XRD), dispersion[734], fluorescence[735,736] (XRF), and photoelectron

spectroscopy[737,738] (XPS). Quantitative advantages of electron microscopes can include higher resolution and depth of field, and lower radiation damage than light microscopes[739]. In addition, electron microscopes can record images, enabling visual interpretation of complex structures that may otherwise be intractable. This section will briefly introduce varieties of electron microscopes, simulation software, and how electron microscopes can interface with ANNs.

3.1 Microscopes

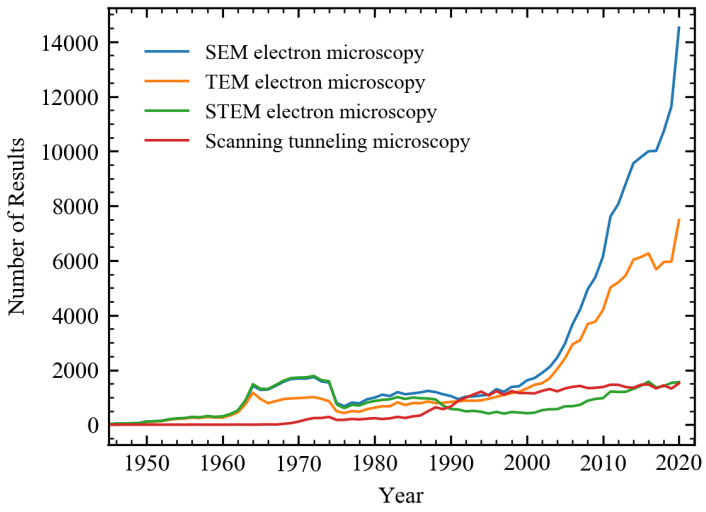

Figure 6. Numbers of results per year returned by Dimensions.ai abstract searches for SEM, TEM, STEM, STM and REM qualitate their popularities. The number of results for 2020 is extrapolated using the mean rate before 14th July 2020.

There are a variety of electron microscopes that use different illumination mechanisms. For example, reflection electron microscopy[740,741] (REM), scanning electron microscopy[742,743] (SEM), scanning transmission electron microscopy[744,745] (STEM), scanning tunneling microscopy[746,747] (STM), and transmission electron microscopy[748–750] (TEM). To roughly gauge popularities of electron microscope varieties, we performed abstract searches with Dimenions.ai[640,751–753] for their abbreviations followed by "electron microscopy" e.g. "REM electron microscopy". Numbers of results per year in fig. 6 qualitate that popularity increases in order REM, STM, STEM, TEM, then SEM. It may be tempting to attribute the popularity of SEM over TEM to the lower cost of SEM[754], which increases accessibility. However, a range of considerations influence the procurement of electron microscopes[755] and hourly pricing at universities[756–760] is similar for SEM and TEM.

In SEM, material surfaces are scanned by sequential probing with a beam of electrons, which are typically accelerated to 0.2-40 keV. The SEM detects quanta emitted from where the beam interacts with the sample. Most SEM imaging uses low-energy secondary electrons. However, reflection electron microscopy[740,741] (REM) uses elastically backscattered electrons and is often complimented by a combination of reflection high-energy electron diffraction[761–763] (RHEED), reflection high-energy electron loss spectroscopy[764,765] (RHEELS) and spin-polarized low-energy electron microscopy[766–768] (SPLEEM). Some SEMs also detect Auger electrons[769,770]. To enhance materials characterization, most SEMs also detect light. The most common light detectors are for cathodoluminescence and energy dispersive r-ray[771,772] (EDX) spectroscopy. Nonetheless, some SEMs also detect Bremsstrahlung radiation[773].

Alternatively, TEM and STEM can detect electrons transmitted through specimens. In conventional TEM, a single region is exposed to a broad electron beam. In contrast, STEM uses a fine electron beam to probe a series of discrete probing locations. Typically, electrons are accelerated across a potential difference to kinetic energies, E_k, of 80-300 keV. Electrons also have rest energy $E_e = m_e c^2$, where m_e is electron rest mass and c is the speed of light. The total energy, $E_t = E_e + E_k$, of free electrons is related to their rest mass energy by a Lorentz factor, γ,

$$E_t = \gamma m_e c^2 , \tag{1}$$

$$\gamma = (1 - v^2/c^2)^{1/2} , \tag{2}$$

where v is the speed of electron propagation in the rest frame of an electron microscope. Electron kinetic energies in TEM and STEM are comparable to their rest energy, $E_e = 511$ keV[774], so relativistic phenomena[775,776] must be considered to accurately describe their dynamics.

Electrons exhibit wave-particle duality[338,339]. Thus, in an ideal electron microscope, the maximum possible detection angle, θ, between two point sources separated by a distance, d, perpendicular to the electron propagation direction is diffraction-limited. The resolution limit for imaging can be quantified by Rayleigh's criterion[777–779]

$$\theta \simeq 1.22\frac{\lambda}{d}, \tag{3}$$

where resolution increases with decreasing wavelength, λ. Electron wavelength increases with increasing accelerating voltage, as described by the relativistic de Broglie relation[780–782],

$$\lambda = hc\left(E_k^2 + 2E_eE_k\right)^{-1/2}, \tag{4}$$

where h is Planck's constant[774]. Electron wavelengths for typical acceleration voltages tabulated by JEOL are in picometres[783]. In comparison, Cu K-α x-rays, which are often used for XRD, have wavelengths near 0.15 nm[784]. In theory, electrons can therefore achieve over 100× higher resolution than x-rays. Electrons and x-rays are both ionizing; however, electrons often do less radiation damage to thin specimens than x-rays[739]. Tangentially, TEM and STEM often achieve over 10 times higher resolution than SEM[785] as transmitted electrons in TEM and STEM are easier to resolve than electrons returned from material surfaces in SEM.

In practice, TEM and STEM are also limited by incoherence[786–788] introduced by inelastic scattering, electron energy spread, and other mechanisms. TEM and STEM are related by an extension of Helmholtz reciprocity[789,790] where the source plane in a TEM corresponds to the detector plane in a STEM[791], as shown in fig. 5. Consequently, TEM coherence is limited by electron optics between the specimen and image, whereas STEM coherence is limited by the illumination system. For conventional TEM and STEM imaging, electrons are normally incident on a specimen[792]. Advantages of STEM imaging can include higher contrast and resolution than TEM imaging, and lower radiation damage[793]. Following, STEM is increasing being favored over TEM for high-resolution studies. However, we caution that definitions of TEM and STEM resolution can be disparate[794].

In addition to conventional imaging, TEM and STEM include a variety of operating modes for different applications. For example, TEM operating configurations include electron diffraction[795]; convergent beam electron diffraction[796–798] (CBED); tomography[799–806]; and bright field[749,807–809], dark field[749,809] and annular dark field[810] imaging. Similarly, STEM operating configurations include differential phase contrast[811–814]; tomography[799,801–803]; and bright field[815,816] or dark field[817] imaging. Further, electron cameras[818,819] are often supplemented by secondary signal detectors. For example, elemental composition is often mapped by EDX spectroscopy, electron energy loss spectroscopy[820,821] (EELS) or wavelength dispersive spectroscopy[822,823] (WDS). Similarly, electron backscatter diffraction[824–826] (EBSD) can detect strain[827,828]

3.2 Contrast Simulation

The propagation of electron wavefunctions though electron microscopes can be described by wave optics[132]. Following, the most popular approach to modelling measurement contrast is multislice simulation[829,830], where an electron wavefunction is iteratively perturbed as it travels through a model of a specimen. Multislice software for electron microscopy includes ACEM[830–832], clTEM[833,834], cudaEM[835], Dr. Probe[836,837], EMSoft[838,839], JEMS[840], JMULTIS[841], MULTEM[842–844], NCEMSS[845,846], NUMIS[847], Prismatic[848–850], QSTEM[851], SimulaTEM[852], STEM-CELL[853], Tempas[854], and xHREM[855–860]. We find that most multislice software is a recreation and slight modification of common functionality, possibly due to a publish-or-perish culture in academia[861–863]. Bloch-wave simulation[830,864–868] is an alternative to multislice simulation that can reduce computation time and memory requirements for crystalline materials[869].

3.3 Automation

Most modern electron microscopes support Gatan Microscopy Suite (GMS) Software[870]. GMS enables electron microscopes to be programmed by DigitalMicrograph Scripting, a propriety Gatan programming language akin to a simplified version of C++. A variety of DigitalMicrograph scripts, tutorials and related resources are available from Dave Mitchell's DigitalMicrograph Scripting Website[668,669], FELMI/ZFE's Script Database[871] and Gatan's Script library[872]. Some electron microscopists also provide DigitalMicrograph scripting resources on their webpages[873–875]. However, DigitalMicrograph scripts are slow insofar that they are interpreted at runtime, and there is limited native functionality for parallel and distributed computing. As a result, extensions to DigitalMicrograph scripting are often developed in other programming languages that offer more functionality.

Historically, most extensions were developed in C++[876]. This was problematic as there is limited documentation, the standard approach used outdated C++ software development kits such as Visual Studio 2008, and programming expertise required to build functions that interface with DigitalMicrograph scripts limits accessibility. To increase accessibility, recent versions of GMS now support python[877]. This is convenient as it enables ANNs developed with python to readily interface with

electron microscopes. For ANNs developed with C++, users have the option to either create C++ bindings for DigitalMicrograph script or for python. Integrating ANNs developed in other programming languages is more complicated as DigitalMicrograph provides almost no support. However, that complexity can be avoided by exchanging files from DigitalMicrograph script to external libraries via a random access memory (RAM) disk[878] or secondary storage[879].

Increasing accessibility, there are collections of GMS plugins with GUIs for automation and analysis[873–875,880]. In addition, various individual plugins are available[881–885]. Some plugins are open source, so they can be adapted to interface with ANNs. However, many high-quality plugins are proprietary and closed source, limiting their use to automation of data collection and processing. Plugins can also be supplemented by a variety of libraries and interfaces for electron microscopy signal processing. For example, popular general-purpose software includes ImageJ[886], pycroscopy[515,516] and HyperSpy[517,518]. In addition, there are directories for tens of general-purpose and specific electron microscopy programs[887–889].

4 Components

Most modern ANNs are configured from a variety of DLF components. To take advantage of hardware accelerators[58], most ANNs are implemented as sequences of parallelizable layers of tensor operations[890]. Layers are often parallelized across data and may be parallelized across other dimensions[891]. This section introduces popular nonlinear activation functions, normalization layers, convolutional layers, and skip connections. To add insight, we provide comparative discussion and address some common causes of confusion.

4.1 Nonlinear Activation

DNNs need multiple layers to be universal approximators[35–43]. Nonlinear activation functions[892,893] are therefore essential to DNNs as successive linear layers can be contracted to a single layer. Activation functions separate artificial neurons, similar to biological neurons[894]. To learn efficiently, most DNNs are tens or hundreds of layers deep[45,895–897]. High depth increases representational capacity[45], which can help training by gradient descent as DNNs evolve as linear models[898] and nonlinearities can create suboptimal local minima where data cannot be fit by linear models[899]. There are infinitely many possible activation functions. However, most activation functions have low polynomial order, similar to physical Hamiltonians[45].

Most ANNs developed for electron microscopy are for image processing, where the most popular nonlinearities are rectifier linear units[900,901] (ReLUs). The ReLU activation, $f(x)$, of an input, x, and its gradient, $\partial_x f(x)$, are

$$f(x) = \max(0,x) \tag{5a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \tag{5b}$$

Popular variants of ReLUs include Leaky ReLU[902],

$$f(x) = \max(\alpha x,x) \tag{6a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} \alpha, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \tag{6b}$$

where α is a hyperparameter, parametric ReLU[20] (PreLU) where α is a learned parameter, dynamic ReLU where α is a learned function of inputs[903], and randomized leaky ReLU[904] (RReLU) where α is chosen randomly. Typically, learned PreLU α are higher the nearer a layer is to ANN inputs. Motivated by limited comparisons that do not show a clear performance difference between ReLU and leaky ReLU[905], some blogs[906] argue against using leaky ReLU due to its higher computational requirements and complexity. However, an in-depth comparison found that leaky ReLU variants consistently slightly outperform ReLU[904]. In addition, the non-zero gradient of leaky ReLU for $x \leq 0$ prevents saturating, or "dying", ReLU[907–909], where the zero gradient of ReLUs stops learning.

There are a variety of other piecewise linear ReLU variants that can improve performance. For example, ReLUh activations are limited to a threshold[910], h, so that

$$f(x) = \min(\max(0,x),h) \tag{7a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } 0 < x \leq h \\ 0, & \text{if } x > h \end{cases} \tag{7b}$$

Thresholds near $h = 6$ are often effective, so popular choice is ReLU6. Another popular activation is concatenated ReLU[911] (CRELU), which is the concatenation of ReLU(x) and ReLU$(-x)$. Other ReLU variants include adaptive convolutional[912], bipolar[913], elastic[914], and Lipschitz[915] ReLUs. However, most ReLU variants are uncommon as they are more complicated than ReLU and offer small, inconsistent, or unclear performance gains. Moreover, it follows from the universal approximator theorems[35–43] that disparity between ReLU and its variants approaches zero as network depth increases.

In shallow networks, curved activation functions with non-zero Hessians often accelerate convergence and improve performance. A popular activation is the exponential linear unit[916] (ELU),

$$f(x) = \begin{cases} \alpha(\exp(x) - 1), & \text{if } x \leq 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{8a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} \alpha \exp(x), & \text{if } x \leq 0 \\ 1, & \text{if } x \geq 0 \end{cases} \tag{8b}$$

where α is a learned parameter. Further, a scaled ELU[917] (SELU),

$$f(x) = \begin{cases} \lambda\alpha(\exp(x) - 1), & \text{if } x \leq 0 \\ \lambda x, & \text{if } x \geq 0 \end{cases} \tag{9a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} \lambda\alpha \exp(x), & \text{if } x \leq 0 \\ \lambda, & \text{if } x \geq 0 \end{cases} \tag{9b}$$

with fixed $\alpha = 1.67326$ and scale factor $\lambda = 1.0507$ can be used to create self-normalizing neural networks (SNNs). A SNN cannot be derived from ReLUs or most other activation functions. Activation functions with curvature are especially common in ANNs with only a couple of layers. For example, activation functions in radial basis function (RBF) networks[918–921], which are efficient universal approximators, are often Gaussians, multiquadratics, inverse multiquadratics, or square-based RBFs[922]. Similarly, support vector machines[923–925] (SVMs) often use RBFs, or sigmoids,

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{10a}$$

$$\frac{\partial f(x)}{\partial x} = f(x)\,(1 - f(x)) \tag{10b}$$

Sigmoids can also be applied to limit the support of outputs. Unscaled, or "logistic", sigmoids are often denoted $\sigma(x)$ and are related to tanh by $\tanh(x) = 2\sigma(2x) - 1$. To avoid expensive $\exp(-x)$ in the computation of tanh, we recommend K-tanH[926], LeCun tanh[927] or piecewise linear approximation[928,929].

The activation functions introduced so far are scalar functions than can be efficiently computed in parallel for each input element. However, functions of vectors, $\mathbf{x} = \{x_1, x_2, ...\}$, are also popular. For example, softmax activation[930],

$$f(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\text{sum}(\exp(\mathbf{x}))} \tag{11a}$$

$$\frac{f(\mathbf{x})}{\partial x_j} = \sum_i f(\mathbf{x})_i(\delta_{ij} - f(\mathbf{x})_j) \tag{11b}$$

is often applied before computing cross-entropy losses for classification networks. Similarly, Ln vector normalization,

$$f(\mathbf{x}) = \frac{\mathbf{x}}{||\mathbf{x}||_n} \tag{12a}$$

$$\frac{f(\mathbf{x})}{\partial x_j} = \frac{1}{||\mathbf{x}||_n}\left(1 - \frac{x_j^n}{||\mathbf{x}||_n^n}\right) \tag{12b}$$

is often applied to n-dimensional vectors to ensure that they lie on a unit n-sphere[337]. Finally, max pooling[931,932],

$$f(\mathbf{x}) = \max(\mathbf{x}) \tag{13a}$$

$$\frac{f(\mathbf{x})}{\partial x_j} = \begin{cases} 1, & \text{if } j = \text{argmax}(\mathbf{x}) \\ 0, & \text{if } j \neq \text{argmax}(\mathbf{x}) \end{cases} \tag{13b}$$

is another popular multivariate activation function that is often used for downsampling. However, max pooling has fallen out of favor as it is often outperformed by strided convolutional layers[933]. Other vector activation functions include squashing nonlinearities for dynamic routing by agreement in capsule networks[934] and cosine similarity[935].

There is a range of other activation functions that are not detailed here for brevity. Further, finding new activation functions

is an active area of research[936,937]. Notable variants include choosing activation functions from a set before training[938,939] and learning activation functions[938,940–943]. Activation functions can also encode probability distributions[944–946] or include noise[929]. Finally, there are a variety of other deterministic activation functions[937,947]. In electron microscopy, most ANNs enable new or enhance existing applications. Subsequently, we recommend using computationally efficient and established activation functions unless there is a compelling reason to use a specialized activation function.

4.2 Normalization

Normalization[948–950] standardizes signals, which can accelerate convergence by gradient descent and improve performance. Batch normalization[951–956] is the most popular normalization layer in image processing DNNs trained with minibatches of N examples. Technically, a "batch" is an entire training dataset and a "minibatch" is a subset; however, the "mini" is often omitted where meaning is clear from context. During training, batch normalization applies a transform,

$$\mu_B = \frac{1}{N}\sum_{i=1}^{N} x_i\,, \tag{14}$$

$$\sigma_B^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i-\mu_B)^2\,, \tag{15}$$

$$\hat{\mathbf{x}} = \frac{\mathbf{x}-\mu_B}{(\sigma_B^2+\varepsilon)^{1/2}}\,, \tag{16}$$

$$\mathrm{BatchNorm}(\mathbf{x}) = \gamma\hat{\mathbf{x}}+\beta\,, \tag{17}$$

where $\mathbf{x} = \{x_1,...,x_N\}$ is a batch of layer inputs, γ and β are a learnable scale and shift, and ε is a small constant added for numerical stability. During inference, batch normalization applies a transform,

$$\mathrm{BatchNorm}(\mathbf{x}) = \frac{\gamma}{(\mathrm{Var}[x]+\varepsilon)^{1/2}}\mathbf{x} + \left(\beta - \frac{\gamma \mathrm{E}[x]}{(\mathrm{Var}[x]+\varepsilon)^{1/2}}\right)\,, \tag{18}$$

where E[x] and Var[x] are expected batch means and variances. For convenience, E[x] and Var[x] are often estimated with exponential moving averages that are tracked during training. However, E[x] and Var[x] can also be estimated by propagating examples through an ANN after training.

Increasing batch size stabilizes learning by averaging destabilizing loss spikes over batches[251]. Batched learning also enables more efficient utilization of modern hardware accelerators. For example, larger batch sizes improve utilization of GPU memory bandwidth and throughput[379,957,958]. Using large batches can also be more efficient than many small batches when distributing training across multiple CPU clusters or GPUs due to communication overheads. However, the performance benefits of large batch sizes can come at the cost of lower test accuracy as training with large batches tends to converge to sharper minima[959,960]. As a result, it often best not to use batch sizes higher than $N \approx 32$ for image classification[961]. However, learning rate scaling[529] and layer-wise adaptive learning rates[962] can increase accuracy of training with fixed larger batch sizes. Batch size can also be increased throughout training without compromising accuracy[963] to exploit effective learning rates being inversely proportional to batch size[529,963]. Alternatively, accuracy can be improved by creating larger batches from replicated instances of training inputs with different data augmentations[964].

There are a few caveats to batch normalization. Originally, batch normalization was applied before activation[952]. However, applying batch normalization after activation often slightly improves performance[965,966]. In addition, training can be sensitive to the often-forgotten ε hyperaparameter[967] in eqn. 16. Typically, performance decreases as epsilon is increased above $\varepsilon \approx 0.001$; however, there is a sharp increase in performance around $\varepsilon = 0.01$ on ImageNet. Finally, it is often assumed that batches are representative of the training dataset. This is often approximated by shuffling training data to sample independent and identically distributed (i.i.d.) samples. However, performance can often be improved by prioritizing sampling[968,969]. We observe that batch normalization is usually effective if batch moments, μ_B and σ_B, have similar values for every batch.

Batch normalization is less effective when training batch sizes are small, or do not consist of independent samples. To improve performance, standard moments in eqn. 16 can be renormalized[970] to expected means, μ, and standard deviations, σ,

$$\hat{\mathbf{x}} \leftarrow r\hat{\mathbf{x}} + d\,, \tag{19}$$

$$r = \mathrm{clip}_{[1/r_{\max}, r_{\max}]}\left(\frac{\sigma_B}{\sigma}\right)\,, \tag{20}$$

$$d = \mathrm{clip}_{[-d_{\max}, d_{\max}]}\left(\frac{\mu_B-\mu}{\sigma}\right)\,, \tag{21}$$

where gradients are not backpropagated with respect to (w.r.t.) the renormalization parameters, r and d. Moments, μ and σ are tracked by exponential moving averages and clipping to $r_{\max}$ and $d_{\max}$ improves learning stability. Usually, clipping values are increased from starting values of $r_{\max} = 1$ and $d_{\max} = 0$, which correspond to batch normalization, as training progresses. Another approach is virtual batch normalization[971] (VBN), which estimates μ and σ from a reference batch of samples and does not require clipping. However, VBN is computationally expensive as it requires computing a second batch of statistics at every training iteration. Finally, online[972] and streaming[950] normalization enable training with small batch sizes by replace μ_B and σ_B in eqn. 16 with their exponential moving averages.

There are alternatives to the L_2 batch normalization of eqns. 14-18 that standardize to different Euclidean norms. For example, L_1 batch normalization[973] computes

$$s_1 = \frac{1}{N}\sum_{i=1}^{N} |x_i - \mu_B| \,, \tag{22}$$

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu_B}{C_{L_1} s_1} \,, \tag{23}$$

where $C_{L_1} = (\pi/2)^{1/2}$. Although the C_{L_1} factor could be learned by ANNs parameters, its inclusion accelerates convergence of the original implementation of L_1 batch normalization[973]. Another alternative is L_∞ batch normalization[973], which computes

$$s_\infty = \text{mean}(\text{top}_k(|\mathbf{x} - \mu_B|)) \,, \tag{24}$$

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu_B}{C_{L_\infty} s_\infty} \,, \tag{25}$$

where C_{L_∞} is a scale factor, and $\text{top}_k(\mathbf{x})$ returns the k highest elements of $\mathbf{x}$. Hoffer *et al* suggest $k = 10$[973]. Some L_1 batch normalization proponents claim that L_1 batch normalization outperforms[951] or achieves similar performance[973] to L_2 batch normalization. However, we found that L_1 batch normalization often lowers performance in our experiments. Similarly, L_∞ batch normalization often lowers performance[973]. Overall, L_1 and L_∞ batch normalization do not appear to offer a substantial advantage over L_2 batch normalization.

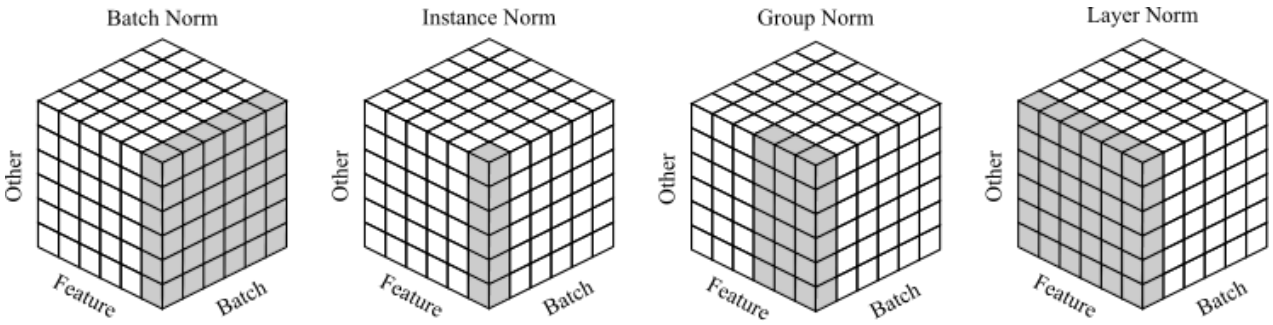

Figure 7. Visual comparison of various normalization methods highlighting regions that they normalize. Regions can be normalized across batch, feature and other dimensions, such as height and width.

A variety of layers normalize samples independently, including layer, instance, and group normalization. They are compared with batch normalization in fig. 7. Layer normalization[974,975] is a transposition of batch normalization that is computed across feature channels for each training example, instead of across batches. Batch normalization is ineffective in RNNs; however, layer normalization of input activations often improves accuracy[974]. Instance normalization[976] is an extreme version of layer normalization that standardizes each feature channel for each training example. Instance normalization was developed for style transfer and makes ANNs insensitive to input image contrast. Group normalization[977] is intermediate to instance and layer normalization insofar that it standardizes groups of channels for each training example.

The advantages of a set of multiple different normalization layers, Ω, can be combined by switchable normalization[978,979], which standardizes to

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \sum_{z \in \Omega} \lambda_z^\mu \mu_z}{\sum_{z \in \Omega} \lambda_z^\sigma \sigma_z} \,, \tag{26}$$

where μ_z and σ_z are means and standard deviations computed by normalization layer z. Their respective importance ratios, λ_z^{μ} and λ_z^{σ}, are trainable parameters that are softmax activated to sum to unity. Combining batch and instance normalization statistics[980] outperforms batch normalization for a range of computer vision tasks[980]. However, most layers strongly weighted either batch or instance normalization, with most preferring batch normalization. Interestingly, combining batch, instance and layer normalization statistics[978,979] results in instance normalization being used more heavily in earlier layers, whereas layer normalization was preferred in the later layers, and batch normalization was preferred in the middle. Smaller batch sizes lead to a preference towards layer normalization and instance normalization. Limitingly, using multiple normalization layers increases computation. To limit expense, we therefore recommend either defaulting to batch normalization, or progressively using single instance, batch or layer normalization layers.

A significant limitation of batch normalization is that it is not effective in RNNs. This is a limited issue as most electron microscopists are developing CNNs for image processing. However, we anticipate that RNNs may become more popular in electron microscopy following the increasing popularity of reinforcement learning[981]. In addition to general-purpose alternatives to batch normalization that are effective in RNNs, such as layer normalization, there are a variety of dedicated normalization schemes. For example, recurrent batch normalization[982,983] uses distinct normalization layers for each time step. Alternatively, batch normalized RNNs[984] only have normalization layers between their input and hidden states. Finally, online[972] and streaming[950] normalization are general-purpose solutions that improve the performance of batch normalization in RNNs by applying batch normalization based on a stream of past batch statistics.

Normalization can also standardize trainable weights, $\mathbf{w}$. For example, weight normalization[985],

$$\text{WeightNorm}(\mathbf{w}) = \frac{g}{||\mathbf{w}||_2}\mathbf{w}, \tag{27}$$

decouples the L2 norm, g, of a variable from its direction. Similarly, weight standardization[986] subtracts means from variables and divides them by their standard deviations,

$$\text{WeightStd}(\mathbf{w}) = \frac{\mathbf{w} - \text{mean}(\mathbf{w})}{\text{std}(\mathbf{w})}, \tag{28}$$

similar to batch normalization. Weight normalization often outperforms batch normalization at small batch sizes. However, batch normalization consistently outperforms weight normalization at larger batch sizes used in practice[987]. Combining weight normalization with running mean-only batch normalization can accelerate convergence[985]. However, similar final accuracy can be achieved without mean-only batch normalization at the cost of slower convergence, or with the use of zero-mean preserving activation functions[913,973]. To achieve similar performance to batch normalization, norm-bounded weight normalization[973] can be applied to DNNs with scale-invariant activation functions, such as ReLU. Norm-bounded weight normalization fixes g at initialization to avoid learning instability[973,987], and scales outputs with the final DNN layer.

Limitedly, weight normalization encourages the use of a small number of features to inform activations[988]. To maximize feature utilization, spectral normalization[988],

$$\text{SpectralNorm}(\mathbf{w}) = \frac{\mathbf{w}}{\sigma(\mathbf{w})}, \tag{29}$$

divides tensors by their spectral norms, $\sigma(\mathbf{w})$. Further, spectral normalization limits Lipschitz constants[989], which often improves generative adversarial network[990–993] (GAN) training by bounding backpropagated discriminator gradients[988]. The spectral norm of $\mathbf{v}$ is the maximum value of a diagonal matrix, Σ, in the singular value decomposition[994–997] (SVG),

$$\mathbf{v} = \mathbf{U}\Sigma\mathbf{V}^*, \tag{30}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices of orthonormal eigenvectors for $\mathbf{v}\mathbf{v}^T$ and $\mathbf{v}^T\mathbf{v}$, respectively. To minimize computation, $\sigma(\mathbf{w})$ is often approximated by the power iteration method[998,999],

$$\hat{\mathbf{v}} \leftarrow \frac{\mathbf{w}^{\mathrm{T}}\hat{\mathbf{u}}}{||\mathbf{w}^{\mathrm{T}}\hat{\mathbf{u}}||_2}, \tag{31}$$

$$\hat{\mathbf{u}} \leftarrow \frac{\mathbf{w}\hat{\mathbf{v}}}{||\mathbf{w}\hat{\mathbf{v}}||_2}, \tag{32}$$

$$\sigma(\mathbf{w}) \simeq \hat{\mathbf{u}}^T\mathbf{w}\hat{\mathbf{v}}, \tag{33}$$

where one iteration of eqns. 31-32 per training iteration is usually sufficient.

Parameter normalization can complement or be combined with signal normalization. For example, scale normalization[1000],

$$\text{ScaleNorm}(\mathbf{x}) = \frac{g}{||\mathbf{x}||_2}\mathbf{x}, \tag{34}$$

learns scales, g, for activations, and is often combined with weight normalization[985,1001] in transformer networks. Similarly, cosine normalization[935],

$$\text{CosineNorm}(\mathbf{x}) = \frac{\mathbf{w}}{||\mathbf{w}||_2} \cdot \frac{\mathbf{x}}{||\mathbf{x}||_2}, \tag{35}$$

computes products of L2 normalized parameters and signals. Both scale and cosine normalization can outperform batch normalization.

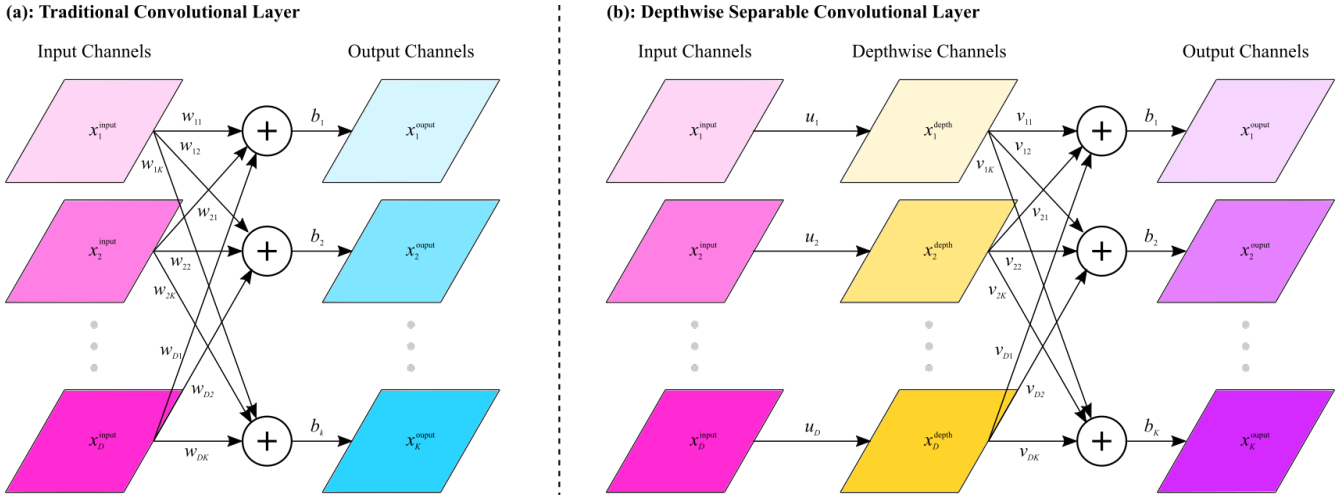

Figure 8. Visualization of convolutional layers. a) Traditional convolutional layer where output channels are sums of biases and convolutions of weights with input channels. b) Depthwise separable convolutional layer where depthwise convolutions compute one convolution with weights for each input channels. Output channels are sums of biases and pointwise convolutions weights with depthwise channels.

4.3 Convolutional Layers

A convolutional neural network[1002–1005] (CNNs) is trained to weight convolutional kernels to exploit local correlations, such as spatial correlations in electron micrographs[220]. Historically, the development of CNNs was inspired by primate visual cortices[1006], where partially overlapping neurons are only stimulated by visual stimuli within their receptive fields. Based on this idea, Fukushima published his Neocognitron[1007–1010] in 1980. Following, convolutional formulations were published by Atlas *et al* in 1988 for a single-layer CNN[1011], and LeCun *et al* in 1998 for a multi-layer CNN[1012,1013]. Following, GPUs were applied to accelerate convolutions in 2010[1014], leading to a breakthrough in classification performance on ImageNet with AlexNet in 2012[67]. Indeed, the deep learning era is often partitioned into before and after AlexNet[17]. Deep CNNs are now ubiquitous. For example, there are review papers on applications of CNNs to action recognition in videos[1015], cytometry[1016], image and video compression[1017,1018], image background subtraction[1019], image classification[262], image style transfer[1020], medical image analysis[320–322,1021–1028], object detection[1029,1030], semantic image segmentation[293,320–322], and text classification[1031].

In general, the convolution of two functions, $f(t)$ and $g(t)$, is

$$(f * g)(x) \coloneqq \int_{s\in\Omega} f(s)g(x-s)\,\mathrm{d}s, \tag{36}$$

and their cross-correlation is

$$(f \circ g)(x) \coloneqq \int_{s\in\Omega} f(s)g(x+s)\,\mathrm{d}s, \tag{37}$$

where integrals have unlimited support, Ω. In a CNN, convolutional layers sum convolutions of feature channels with trainable kernels, as shown in fig. 8. Thus, $f(t)$ and $g(t)$ are discrete functions, and the integrals in eqns. 36-37 can be replaced with

limited summations. Since cross-correlation is equivalent to convolution if the kernel is flipped in every dimension, and CNN kernels are trainable, convolution and cross-correlation is often interchangeable in deep learning. For example, a TensorFlow function named "tf.nn.convolution" computes cross-correlations[1032]. Nevertheless, the difference between convolution and cross-correlation could be source of subtle errors if convolutional layers from a DLF are used in an image processing pipeline with static asymmetric kernels.

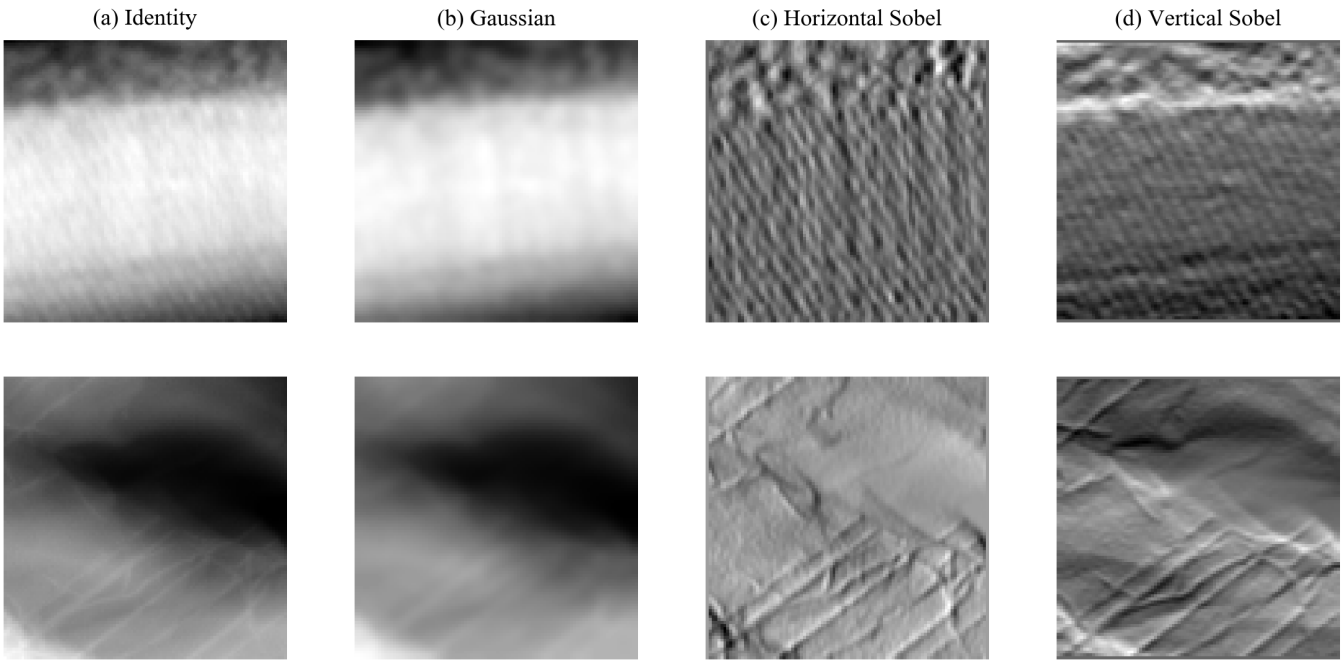

Table 3. A 96×96 electron micrograph a) unchanged, and filtered by b) a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation, c) a 3×3 horizontal Sobel kernel, and a d) 3×3 vertical Sobel kernel. Intensities in a) and b) are in [0, 1], whereas intensities in c) and d) are in [-1, 1].

Filters designed by humans[1033] are often convolved in image processing pipelines. For example, convolutions of electron micrographs with Gaussian and Sobel kernels are shown in table 3. Gaussian kernels compute local averages, blurring images and suppressing high-frequency noise. For example, a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation is

$$\begin{bmatrix}0.1689\\0.2148\\0.2326\\0.2148\\0.1689\end{bmatrix}\begin{bmatrix}0.1689 & 0.2148 & 0.2326 & 0.2148 & 0.1689\end{bmatrix}=\begin{bmatrix}0.0285 & 0.0363 & 0.0393 & 0.0363 & 0.0285\\0.0363 & 0.0461 & 0.0500 & 0.0461 & 0.0363\\0.0393 & 0.0500 & 0.0541 & 0.0500 & 0.0393\\0.0363 & 0.0461 & 0.0500 & 0.0461 & 0.0363\\0.0285 & 0.0363 & 0.0393 & 0.0363 & 0.0285\end{bmatrix}. \tag{38}$$

Alternatives to Gaussian kernels for image smoothing[1034] include mean, median and bilaterial filters. Sobel kernels compute horizontal and vertical spatial gradients that can be used for edge detection[1035]. For example, 3×3 Sobel kernels are

$$\begin{bmatrix}1\\2\\1\end{bmatrix}\begin{bmatrix}1 & 0 & -1\end{bmatrix}=\begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix} \tag{39a}$$

$$\begin{bmatrix}1\\0\\-1\end{bmatrix}\begin{bmatrix}1 & 2 & 1\end{bmatrix}=\begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix} \tag{39b}$$

Alternatives to Sobel kernels offer similar utility, and include extended Sobel[1036], Scharr[1037,1038], Kayyali[1039], Roberts cross[1040] and Prewitt[1041] kernels. Two-dimensional Gaussian and Sobel kernels are examples of linearly separable, or "flattenable", kernels, which can be split into two one-dimensional kernels, as shown in eqns. 38-39b. Kernel separation can decrease computation in convolutional layers by convolving separated kernels in series, and CNNs that only use separable convolutions

are effective[1042–1044]. However, serial convolutions decrease parallelization and separable kernels have fewer degrees of freedom, decreasing representational capacity. Following, separated kernels are usually at least 5×5, and separated 3×3 kernels are unusual. Even-sized kernels, such as 2×2 and 4×4, are rare as symmetric padding is needed to avoid information erosion caused by spatial shifts of feature maps[1045].

A traditional 2D convolutional layer maps inputs, x^{input}, with height H, width, W, and depth, D, to

$$x_{kij}^{\text{output}} = b_k + \sum_{d=1}^{D} \sum_{m=1}^{M} \sum_{n=1}^{N} w_{dkmn} x_{d(i+m-1)(j+n-1)}^{\text{input}}, , i \in [1, H-M+1], j \in [1, W-N+1], \tag{40}$$

where K output channels are indexed by $k \in [1, K]$, is the sum of a bias, b, and convolutions of each input channel with $M \times N$ kernels with weights, w. For clarity, a traditional convolutional layer is visualized in fig. 8a. Convolutional layers for 1D, 3D and higher-dimensional kernels[1046] have a similar form to 2D kernels, where kernels are convolved across each dimension. Most inputs to convolutional layers are padded[1047,1048] to avoid reducing spatial resolutions by kernel sizes, which could remove all resolution in deep networks. Padding is computationally inexpensive and eases implementations of ANNs that would otherwise combine layers with different sizes, such as FractalNet[1049], Inception[1050–1052], NASNet[1053], recursive CNNs[1054,1055], and ResNet[1056]. Pre-padding inputs results in higher performance than post-padding outputs[1057]. Following AlexNet[67], most convolutional layers are padded with zeros for simplicity. Reflection and replication padding achieve similar results to zero padding[1048]. However, padding based on partial convolutions[1058] consistently outperforms other methods[1048].

Convolutional layers are similar to fully connected layers used in multilayer perceptrons[1059,1060] (MLPs). For comparison with eqn. 40, a fully connected, or "dense", layer in a MLP computes

$$x_k^{\text{output}} = b_k + \sum_{d=1}^{D} w_{dk} x_d^{\text{input}}, \tag{41}$$

where every input element is connected to every output element. Convolutional layers reduce computation by making local connections within receptive fields of convolutional kernels, and by convolving kernels rather than using different weights at each input position. Intermediately, fully connected layers can be regularized to learn local connections[1061]. Fully connected layers are sometimes used at the middle of encoder-decoders[1062]. However, such fully connected layers can often be replaced by multiscale atrous, or "holey", convolutions[931] in an atrous spatial pyramid pooling[294,295] (ASPP) module to decrease computation without a significant decrease in performance. Alternatively, weights in fully connected layers can be decomposed into multiple smaller tensors to decrease computation without significantly decreasing performance[1063,1064].

Convolutional layers can perform a variety of convolutional arithmetic[931]. For example, strided convolutions[1065] usually skip computation of outputs that are not at multiples of an integer spatial stride. Most strided convolutional layers are applied throughout CNNs to sequentially decrease spatial extent, and thereby decrease computational requirements. In addition, strided convolutions are often applied at the start of CNNs[527,1050–1052] where most input features can be resolved at a lower resolution than the input. For simplicity and computational efficiency, stride is typically constant within a convolutional layer; however, increasing stride away from the centre of layers can improve performance[1066]. To increase spatial resolution, convolutional layers often use reciprocals of integer strides[1067]. Alternatively, spatial resolution can be increased by combining interpolative upsampling with an unstrided convolutional layer[1068,1069], which can help to minimize output artefacts.

Convolutional layers couple the computation of spatial and cross-channel convolutions. However, partial decoupling of spatial and cross-channel convolutions by distributing inputs across multiple convolutional layers and combining outputs can improve performance. Partial decoupling of convolutions is prevalent in many seminal DNN architectures, including FractalNet[1049], Inception[1050–1052], NASNet[1053]. Taking decoupling to an extreme, depthwise separable convolutions[527,1070,1071] shown in fig. 8 compute depthwise convolutions,

$$x_{dij}^{\text{depth}} = \sum_{m=1}^{M} \sum_{n=1}^{N} u_{dmn} x_{d(i+m-1)(j+n-1)}^{\text{input}}, \tag{42}$$

then compute pointwise 1×1 convolutions for D intermediate channels,

$$x_{kij}^{\text{output}} = b_k + \sum_{d=1}^{D} v_{dk}^{\text{point}} x_{dij}^{\text{depth}}, \tag{43}$$

where K output channels are indexed by $k \in [1, K]$. Depthwise convolution kernels have weights, u, and the depthwise layer is often followed by extra batch normalization before pointwise convolution to improve performance and accelerate convergence[1070]. Increasing numbers of channels with pointwise convolutions can increase accuracy[1070], at the cost of increased

computation. Pointwise convolutions are a special case of traditional convolutional layers in eqn. 40 and have convolution kernel weights, v, and add biases, b. Naively, depthwise separable convolutions require fewer weight multiplications than traditional convolutions[1072,1073]. However, extra batch normalization and serialization of one convolutional layer into depthwise and pointwise convolutional layers mean that depthwise separable convolutions and traditional convolutions have similar computing times[527,1073].

Most DNNs developed for computer vision use fixed-size inputs. Although fixed input sizes are often regarded as an artificial constraint, it is similar animalian vision where there is an effectively constant number of retinal rods and cones[1074–1076]. Typically, the most practical approach to handle arbitrary image shapes is to train a DNN with crops so that it can be tiled across images. In some cases, a combination of cropping, padding and interpolative resizing can also be used. To fully utilize unmodified variable size inputs, a simple is approach to train convolutional layers on variable size inputs. A pooling layer, such as global average pooling, can then be applied to fix output size before fully connected or other layers that might require fixed-size inputs. More involved approaches include spatial pyramid pooling[1077] or scale RNNs[1078]. Typical electron micrographs are much larger than 300×300, which often makes it unfeasible for electron microscopists with a few GPUs to train high-performance DNNs on full-size images. For comparison, Xception was trained on 300×300 images with 60 K80 GPUs for over one month.

The Fourier transform[1079], $\hat{f}(k_1,...,k_N)$, at an N-dimensional Fourier space vector, $\{k_1,...,k_N\}$, is related to a function, $f(x_1,...,x_N)$, of an N-dimensional signal domain vector, $\{x_1,...,x_N\}$, by

$$\hat{f}(k_1,...,k_N) = \left(\frac{|b|}{(2\pi)^{1-a}}\right)^{N/2} \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} f(x_1,...,x_N) \exp(+ibk_1x_i + ... + ibk_Nx_N)\, \mathrm{d}x_1 ... \mathrm{d}x_N\,, \tag{44}$$

$$f(x_1,...,x_N) = \left(\frac{|b|}{(2\pi)^{1+a}}\right)^{N/2} \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} \hat{f}(k_1,...,k_N) \exp(-ibk_1x_i - ... - ibk_Nx_N)\, \mathrm{d}k_1 ... \mathrm{d}k_N\,, \tag{45}$$

where $\pi = 3.141...$, and $i = (-1)^{1/2}$ is the imaginary number. Two parameters, a and b, can parameterize popular conventions that relate the Fourier and inverse Fourier transforms. Mathematica documentation nominates conventions[1080] for general applications (a,b), pure mathematics $(1,-1)$, classical physics $(-1,1)$, modern physics $(0,1)$, systems engineering $(1,-1)$, and signal processing $(0,2\pi)$. We observe that most electron microscopists follow the modern physics convention of $a=0$ and $b=1$; however, the choice of convention is arbitrary and usually does not matter if it is consistent within a project. For discrete functions, Fourier integrals are replaced with summations that are limited to the support of the function.

Discrete Fourier transforms of uniformly spaced inputs are often computed with a fast Fourier transform (FFT) algorithm, which can be parallelized for CPUs[1081] or GPUs[61,1082–1084]. Typically, the speedup of FFTs on GPUs over CPUs is higher for larger signals[1085,1086]. Most popular FFTs are based on the Cooley-Turkey algorithm[1087,1088], which recursively divides FFTs into smaller FFTs. We observe that some electron microscopists consider FFTs to be limited to radix-2 signals that can be recursively halved; however, FFTs can use any combination of factors for the sizes of recursively smaller FFTs. For example, clFFT[1089] FFT algorithms support signal sizes that are any sum of powers of 2, 3, 5, 7, 11 and 13.

Convolution theorems can decrease computation by enabling convolution in the Fourier domain[1090]. To ease notation, we denote the Fourier transform of a signal, $\mathbf{I}$, by $\mathrm{FT}(\mathbf{I})$, and the inverse Fourier transform by $\mathrm{FT}^{-1}(\mathbf{I})$. Following, the convolution theorems for two signals, $\mathbf{I}_1$ and $\mathbf{I}_2$, are[1091]

$$\mathrm{FT}(\mathbf{I}_1 * \mathbf{I}_2) = \mathrm{FT}(\mathbf{I}_1) \cdot \mathrm{FT}(\mathbf{I}_2)\,, \tag{46}$$

$$\mathrm{FT}(\mathbf{I}_1 \cdot \mathbf{I}_2) = \mathrm{FT}(\mathbf{I}_1) * \mathrm{FT}(\mathbf{I}_2)\,, \tag{47}$$

where the signals can be feature channels and convolutional kernels. Fourier domain convolutions, $\mathbf{I}_1 * \mathbf{I}_2 = \mathrm{FT}^{-1}\left(\mathrm{FT}(\mathbf{I}_1) \cdot \mathrm{FT}(\mathbf{I}_2)\right)$, are increasingly efficient, relative to signal domain convolutions, as kernel and image sizes increase[1090]. Indeed, Fourier domain convolutions are exploited to enable faster training with large kernels in Fourier CNNs[1090,1092]. However, Fourier CNNs are rare as most researchers use small 3×3 kernels, following University of Oxford Visual Geometry Group (VGG) CNNs[1093].

4.4 Skip Connections

Residual connections[1056] add a signal after skipping ANN layers, similar to cortical skip connections[1094,1095]. Residuals improve DNN performance by preserving gradient norms during backpropagation[525,1096] and avoiding bad local minima[1097] by smoothing DNN loss landscapes[1098]. In practice, residuals enable DNNs to behave like an ensemble of shallow networks[1099] that learn to iteratively estimate outputs[1100]. Mathematically, a residual layer learns parameters, $\mathbf{w}_l$, of a perturbative function, $f_l(\mathbf{x}_l, \mathbf{w}_l)$, that maps a signal, $\mathbf{x}_l$, at depth l to depth $l+1$,

$$\mathbf{x}_{l+1} = \mathbf{x}_l + f_l(\mathbf{x}_l, \mathbf{w}_l)\,. \tag{48}$$

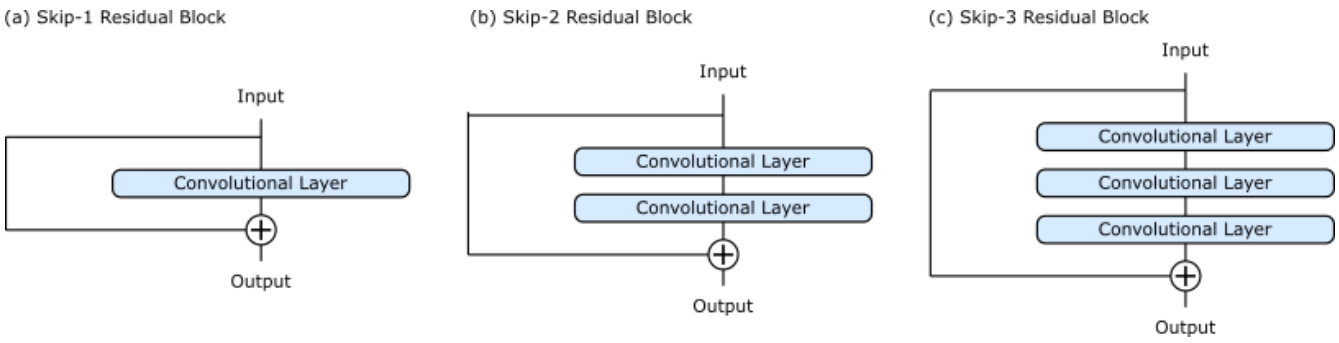

Figure 9. Residual blocks where a) one, b) two, and c) three convolutional layers are skipped. Typically, convolutional layers are followed by batch normalization then activation.

Residuals were developed for CNNs[1056], and examples of residual connections that skip one, two and three convolutional layers are shown in fig. 9. Nonetheless, residuals are also used in MLPs[1101] and RNNs[1102–1104]. Representational capacity of perturbative functions increases as the number of skipped layers increases. As result, most residuals skip two or three layers. Skipping one layer rarely improves performance due to its low representational capacity[1056].

There are a range of residual connection variants that can improve performance. For example, highway networks[1105,1106] apply a gating function to skip connections, and dense networks[1107–1109] use a high number of residual connections from multiple layers. Another example is applying a 1×1 convolutional layer to x_l before addition[527,1056] where $f_l(x_l, w_l)$ spatially resizes or changes numbers of feature channels. However, resizing with norm-preserving convolutional layers[1096] before residual blocks can often improve performance. Finally, long additive[1110] residuals that connect DNN inputs to outputs are often applied to DNNs that learn perturbative functions.

A limitation of preserving signal information with residuals[1111,1112] is that residuals make DNNs learn perturbative functions, which can limit accuracy of DNNs that learn non-perturbative functions if they do not have many layers. Feature channel concatenation is an alternative approach that not perturbative, and that supports combination of layers with different numbers of feature channels. In encoder-decoders, a typical example is concatenating features computed near the start with layers near the end to help resolve output features[294,295,297,305]. Concatenation can also combine embeddings of different[1113,1114] or variants of[354] input features by multiple DNNs. Finally, peephole connections in RNNs can improve performance by using concatenation to combine cell state information with other cell inputs[1115,1116].

5 Architecture

There is a high variety of ANN architectures[3–6] that are trained to minimize losses for a range of applications. Many of the most popular ANNs are also the simplest, and information about them is readily available. For example, encoder-decoder[294–297,490–492] or classifier[262] ANNs usually consist of single feedforward sequences of layers that map inputs to outputs. This section introduces more advanced ANNs used in electron microscopy, including actor-critics, GANs, RNNs, and VAEs. These ANNs share weights between layers or consist of multiple subnetworks. Other notable architectures include recursive CNNs[1054,1055], Network-in-Networks[1117] (NiNs), and transformers[1118,1119]. Although they will not be detailed here, their references may be good starting points for research.

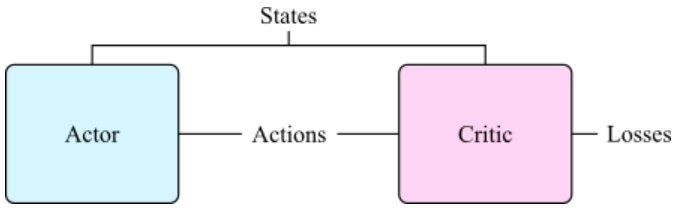

Figure 10. Actor-critic architecture. An actor outputs actions based on input states. A critic then evaluates action-state pairs to predict losses.

5.1 Actor-Critic

Most ANNs are trained by gradient descent using backpropagated gradients of a differentiable loss function c.f. section 6.1. However, some losses are not differentiable. Examples include losses of actors directing their vision[1120,1121], and playing competitive[22] or score-based[1122,1123] computer games. To overcome this limitation, a critic[1124] can be trained to predict differentiable losses from action and state information, as shown in fig. 10. If the critic does not depend on states, it is a surrogate loss function[1125,1126]. Surrogates are often fully trained before actor optimization, whereas critics that depend on actor-state pairs are often trained alongside actors to minimize the impact of catastrophic forgetting[1127] by adapting to changing actor policies and experiences. Alternatively, critics can be trained with features output by intermediate layers of actors to generate synthetic gradients for backpropagation[1128].

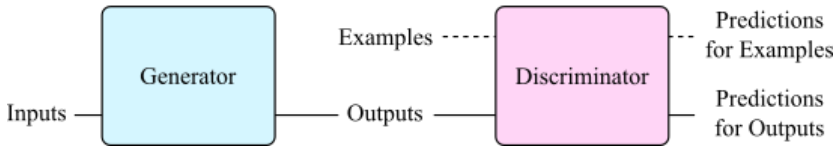

Figure 11. Generative adversarial network architecture. A generator learns to produce outputs that look realistic to a discriminator, which learns to predict whether examples are real or generated.

5.2 Generative Adversarial Network

Generative adversarial networks[990–993] (GANs) consist of discriminator and generator subnetworks that play an adversarial game, as shown in fig. 11. Generators learn to generate outputs that look realistic to a discriminator, whereas the discriminator learns to predict whether examples are real or generated. Most GANs are developed to generate visual media with realistic characteristics. For example, partial STEM images infilled with a GAN are less blurry than images infilled with a non-adversarial generator trained to minimize MSEs[192]. Alternatively, computationally inexpensive loss functions engineered by humans, such as SSIM[1129] and Sobel losses[220], can improve generated output realism. However, it follows from the universal approximator theorems[35–43] that training with ANN discriminators can often yield more realistic outputs.

There are many popular GAN loss functions and regularization mechanisms[1130–1134]. Originally, GANs were trained to minimize logarithmic discriminator, D, and generator, G, losses[1135],

$$L_D = -\log D(\mathbf{x}) - \log(1 - D(G(\mathbf{z}))), \tag{49}$$

$$L_G = \log(1 - D(G(\mathbf{z}))), \tag{50}$$

where $\mathbf{z}$ are generator inputs, $G(\mathbf{z})$ are generated outputs, and $\mathbf{x}$ are example outputs. Discriminators predict labels, $D(\mathbf{x})$ and $D(G(\mathbf{z}))$, where target labels are 0 and 1 for generated and real examples, respectively. Limitedly, logarithmic losses are numerically unstable for $D(\mathbf{x}) \to 0$ or $D(G(\mathbf{z})) \to 1$, as the denominator, $f(x)$, in $\partial_x \log f(x) = \partial_x f(x)/f(x)$ vanishes. In addition, discriminators must be limited to $D(\mathbf{x}) > 0$ and $D(G(\mathbf{z})) < 1$, so that logarithms are not complex. To avoid these issues, we recommend training discriminators with squared difference losses[1136,1137],

$$L_D = (D(\mathbf{x}) - 1)^2 + D(G(\mathbf{z}))^2, \tag{51}$$

$$L_G = (D(G(\mathbf{z})) - 1)^2. \tag{52}$$

Nevertheless, there are a variety of other alternatives to logarithmic loss functions that are also effective[1130,1131].

A variety of methods have been developed to improve GAN training[971,1138]. The most common issues are catastrophic forgetting[1127] of previous learning, and mode collapse[1139] where generators only output examples for a subset of a target domain. Mode collapse often follows discriminators becoming Lipschitz discontinuous. Wasserstein GANs[1140] avoid mode collapse by clipping trainable variables, albeit often at the cost of 5-10 discriminator training iterations per generator training iteration. Alternatively, Lipschitz continuity can be imposed by adding a gradient penalty[1141] to GAN losses, such as differences of L2 norms of discriminator gradients from unity,

$$\tilde{x} = G(\mathbf{z}), \tag{53}$$

$$\hat{\mathbf{x}} = \varepsilon \mathbf{x} + (1 - \varepsilon)\tilde{\mathbf{x}}, \tag{54}$$

$$L_D = D(\tilde{\mathbf{x}}) - D(\mathbf{x}) + \lambda(||\partial_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})||_2 - 1)^2, \tag{55}$$

$$L_G = -D(G(\mathbf{z})), \tag{56}$$

where $\varepsilon \in [0, 1]$ is a uniform random variate, λ weights the gradient penalty, and $\tilde{\mathbf{x}}$ is an attempt to generate x. However, using a gradient penalty introduces additional gradient backpropagation that increases discriminator training time. There are also a variety of computationally inexpensive tricks that can improve training, such as adding noise to labels[971,1051,1142] or balancing discriminator and generator learning rates[337]. These tricks can help to avoid discontinuities in discriminator output distributions that can lead to mode collapse; however, we observe that these tricks do not reliably stabilize GAN training.

Instead, we observe that spectral normalization[988] reliably stabilizes GAN discriminator training in our electron microscopy research[192,193,337]. Spectral normalization controls Lipschitz constants of discriminators by fixing the spectral norms of their weights, as introduced in section 4.2. Advantages of spectral normalization include implementations based on the power iteration method[998,999] being computationally inexpensive, not adding a regularizing loss function that could detrimentally compete[1143,1144] with discrimination losses, and being effective with one discriminator training iterations per generator training iteration[988,1145]. Spectral normalization is popular in GANs for high-resolution image synthesis, where it is also applied in generators to stabilize training[1146].

There are a variety of GAN architectures[1147]. For high-resolution image synthesis, computation can be decreased by training multiple discriminators to examine image patches at different scales[192,1148]. For domain translation characterized by textural differences, a cyclic GAN[1149,1150] consisting of two GANs can map from one domain to the other and vice versa. Alternatively, two GANs can share intermediate layers to translate inputs via a shared embedding domain[1151]. Cyclic GANs can also be combined with a siamese network[268–270] for domain translation beyond textural differences[1152]. Finally, discriminators can introduce auxiliary losses to train DNNs to generalize to examples from unseen domains[1153–1155].

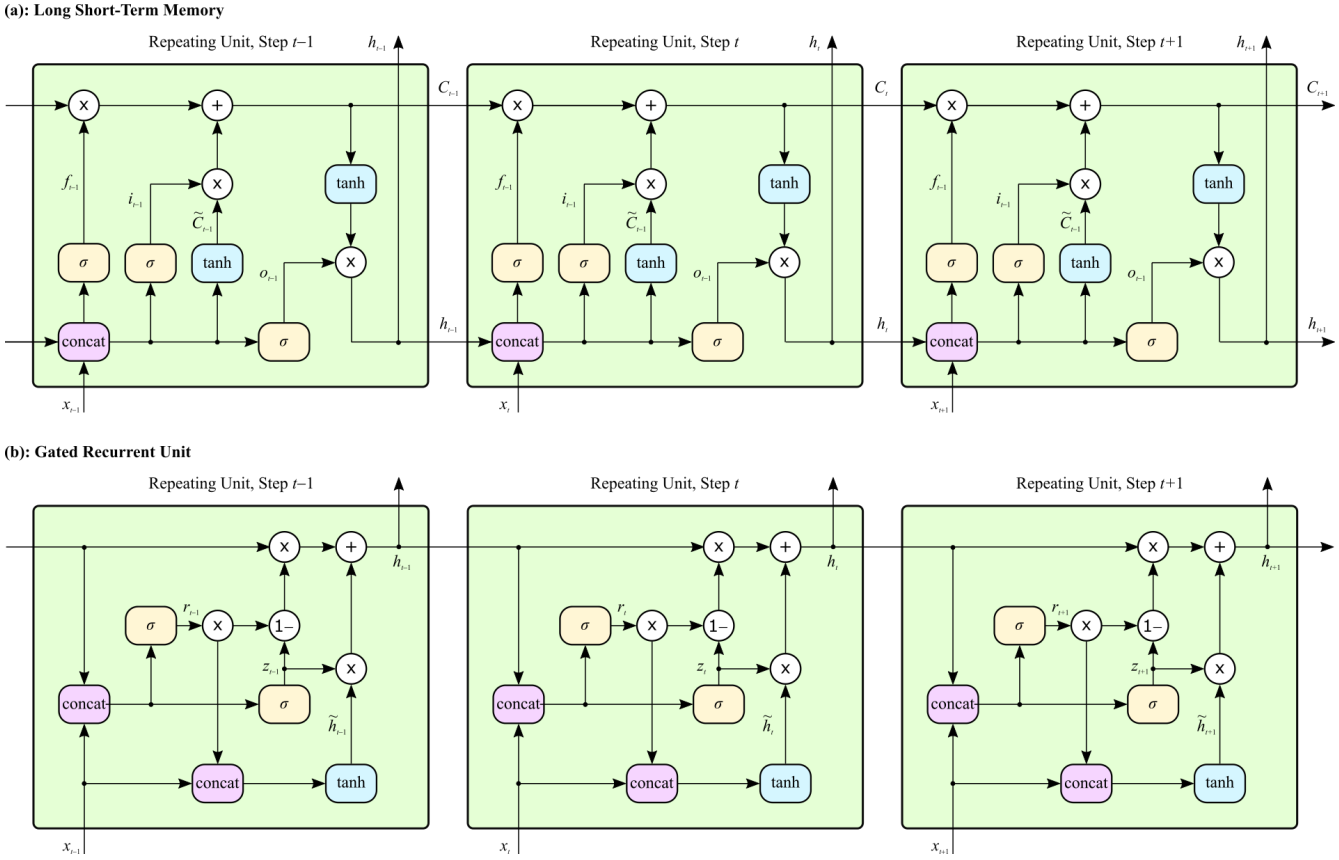

Figure 12. Architectures of recurrent neural networks with a) long short-term memory (LSTM) cells, and b) gated recurrent units (GRUs).

5.3 Recurrent Neural Network

Recurrent neural networks[519–524] reuse an ANN cell to process each step of a sequence. Most RNNs learn to model long-term dependencies by gradient backpropagation through time[1156] (BPTT). Essentially, the ability of RNNs to utilize past experiences enables them to model partially observed and variable length Markov decision processes[1157,1158] (MDPs). Applications of RNNs include directing vision[1120,1121], image captioning[1159,1160], language translation[1161], medicine[73], natural language

processing[1162,1163], playing computer games[22], text classification[1031], and traffic forecasting[1164]. Many RNNs are combined with CNNs to embed visual media[1121] or words[1165,1166], or to process RNN outputs[1167,1168]. RNNs can also be combined with with MLPs[1120], or text embeddings[1169] such as BERT[1169,1170], continuous bag-of-words[1171–1173] (CBOW), doc2vec[1174,1175], GloVe[1176] and word2vec[1171,1177].

The most popular RNNs consist of short-term memory[1178–1181] (LSTM) cells and gated recurrent units[1179,1182–1184] (GRUs). LSTMs and GRUs are popular as they solve the vanishing gradient problem[525,1185,1186] and have consistently high performance[1187–1192]. Their architecture is shown in fig. 12. At step t, an LSTM outputs a hidden state, h_t, and cell state, C_t, given by

$$\mathbf{f}_t = \sigma(\mathbf{w}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f), \tag{57}$$
$$\mathbf{i}_t = \sigma(\mathbf{w}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), \tag{58}$$
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{w}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C), \tag{59}$$
$$\mathbf{C}_t = \mathbf{f}_t \mathbf{C}_{t-1} + \mathbf{i}_t \tilde{\mathbf{C}}_t, \tag{60}$$
$$\mathbf{o}_t = \sigma(\mathbf{w}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o), \tag{61}$$
$$\mathbf{h}_t = \mathbf{o}_t \tanh(\mathbf{C}_t), \tag{62}$$

where $\mathbf{C}_{t-1}$ is the previous cell state, $\mathbf{h}_{t-1}$ is the previous hidden state, $\mathbf{x}_t$ is the step input, and σ is a logistic sigmoid function of eqn. 10a, $[\mathbf{x}, \mathbf{y}]$ is the concatenation of $\mathbf{x}$ and $\mathbf{y}$ channels, and $(\mathbf{w}_f, \mathbf{b}_f)$, $(\mathbf{w}_i, \mathbf{b}_i)$, $(\mathbf{w}_C, \mathbf{b}_C)$ and $(\mathbf{w}_o, \mathbf{b}_o)$ are pairs of weights and biases. A GRU performs fewer computations than an LSTM and does not have separate cell and hidden states,

$$\mathbf{z}_t = \sigma(\mathbf{w}_z \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_z), \tag{63}$$
$$\mathbf{r}_t = \sigma(\mathbf{w}_r \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_r), \tag{64}$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{w}_h \cdot [\mathbf{r}_t \mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_h), \tag{65}$$
$$\mathbf{h}_t = (1 - \mathbf{z}_t)\mathbf{h}_{t-1} + \mathbf{z}_t \tilde{\mathbf{h}}_t, \tag{66}$$

where $(\mathbf{w}_z, \mathbf{b}_z)$, $(\mathbf{w}_r, \mathbf{b}_r)$, and $(\mathbf{w}_h, \mathbf{b}_h)$ are pairs of weights and biases. Minimal gated units (MGUs) can further reduce computation[1193]. A large-scale analysis of RNN architectures for language translation found that LSTMs consistently outperform GRUs[1187]. GRUs struggle with simple languages that are learnable by LSTMs as the combined hidden and cell states of GRUs make it more difficult for GRUs to perform unbounded counting[1191]. However, further investigations found that GRUs can outperform LSTMs on tasks other than language translation[1188], and that GRUs can outperform LSTMs on some datasets[1189,1190,1194]. Overall, LSTM performance is comparable to that of GRUs.

There are a variety of alternatives to LSTM and GRUs. Examples include continuous time RNNs[1195–1199] (CTRNNs), Elman[1200] and Jordan[1201] networks, independently RNNs[1202] (IndRNNs), Hopfield networks[1203], recurrent MLPs[1204] (RMLPs). However, none of the variants offer consistent performance benefits over LSTMs for general sequence modelling. Similarly, augmenting LSTMs with additional connections, such as peepholes[1115,1116] and projection layers[1205], does not consistently improve performance. For electron microscopy, we recommend defaulting to LSTMs as we observe that their performance is more consistently high than performance of other RNNs. However, LSTM and GRU performance is often comparable, so GRUs are also a good choice to reduce computation.

There are a variety of architectures based on RNNs. Popular examples include deep RNNs[1206] that stack RNN cells to increase representational ability, bidirectional RNNs[1207–1210] that process sequences both forwards and in reverse to improve input utilization, and using separate encoder and decoder subnetworks[1182,1211] to embed inputs and generate outputs. Hierarchical RNNs[1212–1216] are more complex models that stack RNNs to efficiently exploit hierarchical sequence information, and include multiple timescale RNNs[1217,1218] (MTRNNs) that operate at multiple sequence lengthscales. Finally, RNNs can be augmented with additional functionality to enable new capabilities. For example, attention[1159,1219–1221] mechanisms can enable more efficient input utilization. Further, creating a neural Turing machine (NTMs) by augmenting a RNN with dynamic external memory[1222,1223] can make it easier for an agent to solve dynamic graphs.

5.4 Autoencoders

Autoencoders[1224–1226] (AEs) learn to efficiently encode inputs, $\mathbf{I}$, without supervision. An AE consists of a encoder, E, and decoder, D, as shown in fig. 13a. Most encoders and decoders are jointly trained[1227] to restore inputs from encodings, $E(\mathbf{I})$, to minimize a MSE loss,

$$L_{\text{AE}} = \text{MSE}(D(E(\mathbf{I})), \mathbf{I}), \tag{67}$$

by gradient descent. In practice, DNN encoders and decoders yield better compression[1225] than linear techniques, such as principal component analysis[1228] (PCA), or shallow ANNs. Indeed, deep AEs can outperform JPEG image compression[1229].

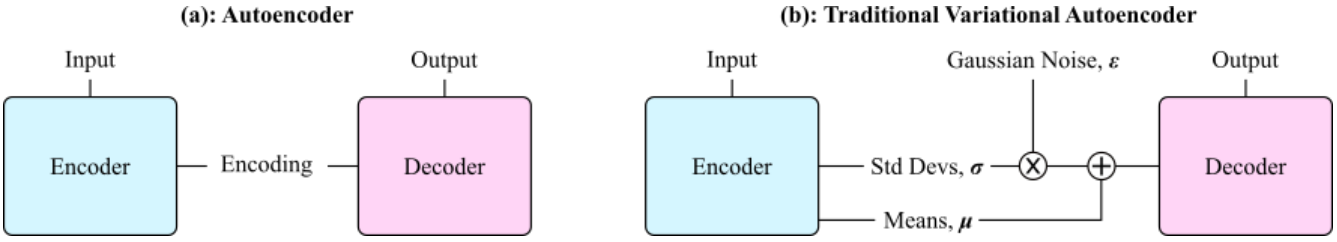

Figure 13. Architectures of autoencoders where an encoder maps an input to a latent space and a decoder learns to reconstruct the input from the latent space. a) An autoencoder encodes an input in a deterministic latent space, whereas a b) traditional variational autoencoder encodes an input as means, μ, and standard deviations, σ, of Gaussian multivariates, $\mu + \sigma \cdot \varepsilon$, where ε is a standard normal multivariate.

Denoising autoencoders[1230–1234] (DAEs) are a popular AE variant that can learn to remove artefacts by artificially corrupting inputs inside encoders. Alternatively, contractive autoencoders[1235,1236] (CAEs) can decrease sensitivity to input values by adding a loss to minimize gradients w.r.t. inputs. Most DNNs that improve electron micrograph signal-to-noise are DAEs.

In general, semantics of AE outputs are pathological functions of encodings. To generate outputs with well-behaved semantics, traditional variational autoencoders[945,1237,1238] (VAEs) learn to encode means, μ, and standard deviations, σ, of Gaussian multivariates. Meanwhile, decoders learn to reconstruct inputs from sampled multivariates, $\mu + \sigma \cdot \varepsilon$, where ε is a standard normal multivariate. Traditional VAE architecture is shown in fig. 13b. Usually, VAE encodings are regularized by adding the Kullback-Leibler (KL) divergence of encodings from standard multinormals to the VAE loss function,

$$L_{\text{VAE}} = \text{MSE}(D(\mu + \sigma \cdot \varepsilon), \mathbf{I}) + \frac{\lambda_{\text{KL}}}{2Bu} \sum_{i=1}^{B} \sum_{j=1}^{u} \mu_{ij}^2 + \sigma_{ij}^2 - \log(\sigma_{ij}^2) - 1\,, \tag{68}$$

where λ_{KL} weights the contribution of the KL divergence loss for a batch size of B, and a latent space with u elements. However, variants of Gaussian regularization can improve clustering[220], and sparse autoencoders[1239–1242] (SAEs) that regularize encoding sparsity can encode more meaningful features. To generate realistic outputs, a VAE can be combined with a GANs to create a VAE-GAN[1243–1245]. Adding a loss to minimize differences between gradients of generated and target outputs is computationally inexpensive alternative that can generate realistic outputs for some applications[220].

A popular application of VAEs is data clustering. For example, VAEs can encode hash tables[1246–1250] for search engines, and we use VAEs as the basis of our electron micrograph search engines[220]. Encoding clusters visualized by tSNE can be labelled to classify data[220], and encoding deviations from clusters can be used for anomaly detection[1251–1255]. In addition, learning encodings with well-behaved semantics enables encodings to be used for semantic manipulation[1255,1256]. Finally, VAEs can be used as generative models to create synthetic populations[1257,1258], develop new chemicals[1259–1262], and synthesize underrepresented data to reduce imbalanced learning[1263].

6 Optimization

Training, testing, deployment and maintenance of machine learning systems is often time-consuming and expensive[1264–1267]. The first step is preparing training data and setting up data pipelines for ANN training and evaluation. Typically, ANN parameters are randomly initialized for optimization by gradient descent, possibly as part of an automatic machine learning algorithm. Reinforcement learning is a special optimization case where the loss is a discounted future reward. During training, ANN components are often regularized to stabilize training, accelerate convergence, or improve performance. Finally, trained models can be streamlined for efficient deployment. This section introduces each step. We find that electron microscopists can be apprehensive about robustness and interpretability of ANNs, so we also provide subsections on model evaluation and interpretation.

6.1 Gradient Descent

Most ANNs are iteratively trained by gradient descent[453,1280–1284], as described by algorithm 1 and shown in fig. 14. To minimize computation, results at intermediate stages of forward propagation, where inputs are mapped to outputs, are often stored in memory. Storing the forwards pass in memory enables backpropagation memoization by sequentially computing gradients w.r.t. trainable parameters. To reduce memory costs for large ANNs, a subset of intermediate forwards pass results can be saved as starting points to recompute other stages during backpropagation[1285,1286]. Alternatively, forward pass computations

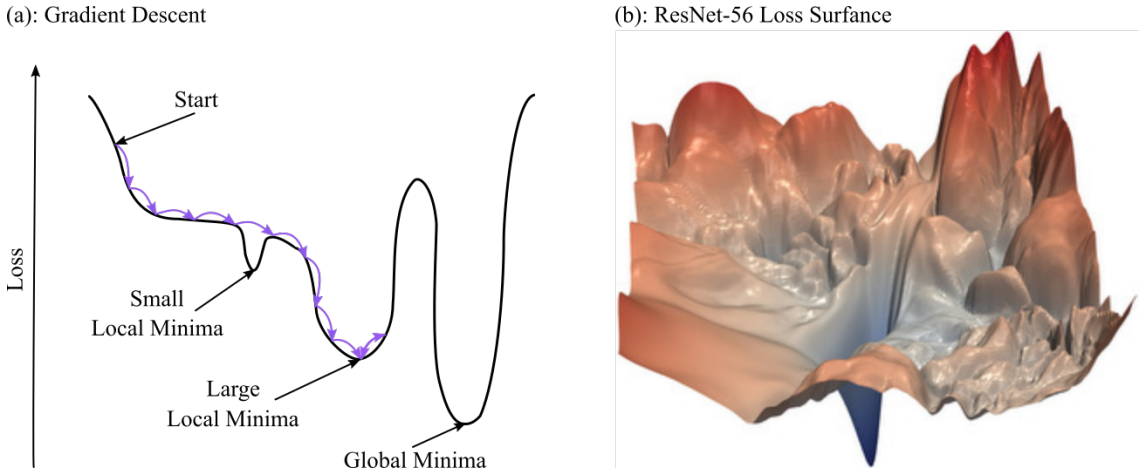

Figure 14. Gradient descent. a) Arrows depict steps across one dimension of a loss landscape as a model is optimized by gradient descent. In this example, the optimizer traverses a small local minimum; however, it then gets trapped in a larger sub-optimal local minimum, rather than reaching the global minimum. b) Experimental DNN loss surface for two random directions in parameter space showing many local minima[1098]. The plot in part b) is reproduced with permission under an MIT license[1268].

Algorithm 1 Optimization by gradient descent.

Initialize a model, $f(\mathbf{x})$, with trainable parameters, θ_1.
for training step $t = 1, T$ **do**
  Forwards propagate a randomly sampled batch of inputs, $\mathbf{x}$, through the model to compute outputs, $\mathbf{y} = f(\mathbf{x})$.
  Compute loss, L_t, for outputs.
  Use the differentiation chain rule[1269] to backpropagate gradients of the loss to trainable parameters, θ_t.
  Apply an optimizer to the gradients to update θ_{t-1} to θ_t.
end for

can be split across multiple devices[1287]. Optimization by gradient descent plausibly models learning in some biological systems[1288]. However, gradient descent is not generally an accurate model of biological learning[1289–1291].

There are many popular gradient descent optimizers for deep learning[1280–1282]. Update rules for eight popular optimizers are summarized in fig. 15. Other optimizers include AdaBound[1292], AMSBound[1292], AMSGrad[1293], Lookahead[1294], NADAM[1295], Nostalgic Adam[1296], Power Gradient Descent[1297], RADAM[1298], and trainable optimizers[1299–1303]. Gradient descent is effective in the high-dimensional optimization spaces of overparameterized ANNs[1304] as the probability of getting trapped in a sub-optimal local minimum decreases as the number of dimensions increases. The simplest optimizer is "vanilla" stochastic gradient descent (SGD), where a trainable parameter perturbation, $\Delta\theta_t = \theta_t - \theta_{t-1}$, is the product of a learning rate, η, and derivative of a loss, L_t, w.r.t. the trainable parameter, $\partial_\theta L_t$. However, vanilla SGD convergence is often limited by unstable parameter oscillations as it a low-order local optimization method[1305]. Further, vanilla SGD has no mechanism to adapt to varying gradient sizes, which vary effective learning rates as $\Delta\theta \propto \partial_\theta L_t$.

To accelerate convergence, many optimizers introduce a momentum term that weights an average of gradients with past gradients[1273,1306,1307]. Momentum-based optimizers in fig. 15 are momentum, Nesterov momentum[1273,1274], quasi-hyperbolic momentum[1276], AggMo[1277], ADAM[1279], and AdaMax[1279]. To standardize effective learning rates for every layer, adaptive optimizers normalize updates based on an average of past gradient sizes. Adaptive optimizers in fig. 15 are RMSProp[1278], ADAM[1279], and AdaMax[1279], which usually result in faster convergance and higher accuracy than other optimizers[1308,1309]. However, adaptive optimizers can be outperformed by vanilla SGD due to overfitting[1310], so some researchers transition from adaptive optimization to vanilla SGD as training progresses[1292]. We recommend adaptive optimization with Nadam[1295], which combines ADAM with Nesterov momentum, as a comparative analysis of select gradient descent optimizers found that it often achieves higher performance than other popular optimizers[1311]. Limitingly, most adaptive optimizers slowly adapt to changing gradient sizes e.g. the recommended value for ADAM β_2 is 0.999[1279]. To prevent learning being destabilized by spikes in gradient sizes, adaptive optimizers can be combined with adaptive learning rate[251,1292] or gradient[1185,1312,1313] clipping.

For non-adaptive optimizers, effective learning rates are likely to vary due to varying magnitudes of gradients w.r.t. trainable parameters. Similarly, learning by biological neurons varies as stimuli usually activate a subset of neurons[1314]. However, all

Vanilla SGD[1270,1271] $[\eta]$

$$\theta_t = \theta_{t-1} - \eta \partial_\theta L_t \tag{69}$$

Momentum[1272] $[\eta, \gamma]$

$$v_t = \gamma v_{t-1} + \eta \partial_\theta L_t \tag{70}$$
$$\theta_t = \theta_{t-1} - v_t \tag{71}$$

Nesterov momentum[1273–1275] $[\eta, \gamma]$

$$\phi = \theta_{t-1} + \eta \gamma v_{t-1} \tag{72}$$
$$v_t = \gamma v_{t-1} + \partial_\theta L_t \tag{73}$$
$$\theta_t = \phi - \eta v_t (1+\gamma) \tag{74}$$

Quasi-hyperbolic momentum[1276] $[\eta, \beta, \nu]$

$$g_t = \beta g_{t-1} + (1-\beta)\partial_\theta L_t \tag{75}$$
$$\theta_t = \theta_{t-1} - \eta (\nu g_t + (1-\nu)\partial_\theta L_t) \tag{76}$$

AggMo[1277] $[\eta, \beta^{(1)}, ..., \beta^{(K)}]$

$$v_t^{(i)} = \beta^{(i)} v_{t-1}^{(i)} - (\partial_\theta L_t) \tag{77}$$
$$\theta_t = \theta_{t-1} + \frac{\eta}{K} \sum_{i=1}^{K} v_t^{(i)} \tag{78}$$

RMSProp[1278] $[\eta, \beta, \varepsilon]$

$$v_t = \beta v_{t-1} + (1-\beta)(\partial_\theta L_t)^2 \tag{79}$$
$$\theta_t = \theta_{t-1} - \frac{\eta}{(v_t + \varepsilon)^{1/2}} \partial_\theta L_t \tag{80}$$

ADAM[1279] $[\eta, \beta_1, \beta_2, \varepsilon]$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\partial_\theta L_t \tag{81}$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\partial_\theta L_t)^2 \tag{82}$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{83}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{84}$$
$$\theta_t = \theta_{t-1} - \frac{\eta}{\hat{v}_t^{1/2} + \varepsilon} \hat{m}_t \tag{85}$$

AdaMax[1279] $[\eta, \beta_1, \beta_2]$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\partial_\theta L_t \tag{86}$$
$$u_t = \max(\beta_2 u_{t-1}, |\partial_\theta L_t|) \tag{87}$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{88}$$
$$\theta_t = \theta_{t-1} - \frac{\eta}{u_t} \hat{m}_t \tag{89}$$

Figure 15. Update rules of various gradient descent optimizers for a trainable parameter, θ_t, at iteration t, gradients of losses w.r.t. the parameter, $\partial_\theta L_t$, and learning rate, η. Hyperparameters are listed in square brackets.

neuron outputs are usually computed for ANNs. Thus, not effectively using all weights to inform decisions is computational inefficient. Further, inefficient weight updates can limit representation capacity, slow convergence, and decrease training stability. A typical example is effective learning rates varying between layers. Following the chain rule, gradients backpropagated to the ith layer of a DNN from its start are

$$\frac{\partial L_t}{\partial \mathbf{x}_i} = \left(\prod_{l=i}^{L-1} \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{x}_l} \right) \frac{\partial L_t}{\partial \mathbf{x}_L}, \tag{90}$$

for a DNN with L layers. Vanishing gradients[525,1185,1186] occur when many layers have $\partial x_{l+1}/\partial x_l \ll 1$. For example, DNNs with logistic sigmoid activations often exhibit vanishing gradients as their maximum gradient is $1/4$ c.f. eqn. 10b. Similarly, exploding gradients[525,1185,1186] occur when many layers have $\partial x_{l+1}/\partial x_l \gg 1$. Adaptive optimizers alleviate vanishing and exploding gradients by dividing gradients by their expected sizes. Nevertheless, it is essential to combine adaptive optimizers with appropriate initialization and architecture to avoid numerical instability.

Optimizers have a myriad of hyperparameters to be initialized and varied throughout training to optimize performance[1315] c.f. fig. 15. For example, stepwise exponentially decayed learning rates are often theoretically optimal[1316]. There are also various heuristics that are often effective, such as using a DEMON decay schedule for an ADAM first moment of the momentum decay rate[1317],

$$\beta_1 = \frac{1 - t/T}{(1 - \beta_{\text{init}}) + \beta_{\text{init}}(1 - t/T)} \beta_{\text{init}}, \tag{91}$$

where β_{init} is the initial value of β_1, t is the iteration number, and T is the final iteration number. Developers often optimize ANN hyperparameters by experimenting with a range of heuristic values. Hyperparameter optimization algorithms[1318–1323] can automate optimizer hyperparameter selection. However, automatic hyperparameter optimizers may not yield sufficient performance improvements relative to well-established heuristics to justify their use, especially in initial stages of development.

Alternatives to gradient descent[1324] are rarely used for parameter optimization as they are not known to consistently improve upon gradient descent. For example, simulated annealing[1325,1326] has been applied to CNN training[1327,1328], and can be augmented with momentum to accelerate convergence in deep learning[1329]. Simulated annealing can also augment gradient descent to improve performance[1330]. Other approaches include evolutionary[1331,1332] and genetic[1333,1334] algorithms, which can be a competitive alternative to deep reinforcement learning where convergence is slow[1335]. Indeed, recent genetic algorithms have outperformed a popular deep reinforcement learning algorithm[1336]. Another direction is to augment genetic algorithms with ANNs to accelerate convergence[1337–1340]. Other alternatives to backpropagation, including direct search[1341], the Moore-Penrose Pseudo Inverse[1342]; particle swarm optimization[1343–1346] (PSO); and echo-state networks[1347–1349] (ESNs) and extreme learning machines[1350–1356] (ELMs), where some randomly initialized weights are never updated.

6.2 Reinforcement Learning

Reinforcement learning[1357–1363] (RL) is where a machine learning system, or "actor", is trained to perform a sequence of actions. Applications include autonomous driving[1364–1366], communications network control[1367,1368], energy and environmental management[1369,1370], playing games[22–27,1122,1371], and robotic manipulation[1372,1373]. To optimize a MDP[1157,1158], a discounted future reward, Q_t, at step t in a MDP with T steps is usually calculated from step rewards, r_t, with Bellman's equation,

$$Q_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}, \tag{92}$$

where $\gamma \in [0, 1)$ discounts future step rewards. To be clear, multiplying Q_t by -1 yields a loss that can be minimized using the methods in section 6.1.

In practice, many MDPs are partially observed or have non-differentiable losses that may make it difficult to learn a good policy from individual observations. However, RNNs can often learn a model of their environments from sequences of observations[1123]. Alternatively, FNNs can be trained with groups of observations that contain more information than individual observations[1122,1371]. If losses are not differentiable, an critic can learn to predict differentiable losses for actor training c.f. section 5.1. Alternatively, actions can be sampled from a differentiable probability distribution[1120,1374] as training losses given by products of losses and sampling probabilities are differentiable. There are also a variety of alternatives to gradient descent introduced at the end of section 6.1 that do not require differentiable loss functions.

There are a variety of exploration strategies for RL[1375,1376]. Adding Ornstein-Uhlenbeck[1377] (OU) noise to actions is effective for continuous control tasks optimized by deep deterministic policy gradients[1122] (DDPG) or recurrent deterministic policy gradients[1123] (RDPG) RL algorithms. Adding Gaussian noise achieves similar performance for optimization by TD3[1378]

or D4PG[1379] RL algorithms. However, a comparison of OU and Gaussian noise across a variety of tasks[1380] found that OU noise usually achieves similar performance to or outperforms Gaussian noise. Similarly, exploration can be induced by adding noise to ANN parameters[1381,1382]. Other approaches to exploration include rewarding actors for increasing action entropy[1382–1384] and intrinsic motivation[1385–1387], where ANNs are incentified to explore actions that they are unsure about.

RL algorithms are often partitioned into online learning[1388,1389], where training data is used as it is acquired; and offline learning[1390,1391], where a static training dataset has already been acquired. However, many algorithms operate in an intermediate regime, where data collected with an online policy is stored in an experience replay[1392–1394] buffer for offline learning. Training data is often sampled at random from a replay. However, prioritizing the replay of data with high losses[969] or data that results in high policy improvements[968] often improves actor performance. A default replay buffer size of around 10^6 examples is often used; however, training is sensitive to replay buffer size[1395]. If the replay is too small, changes in actor policy may destabilize training; whereas if the replay is too large, convergence may be slowed by delays before the actor learns from policy changes.

6.3 Automatic Machine Learning

There are a variety of automatic machine learning[1396–1399] (AutoML) algorithms that can create and optimize ANN architectures and learning policies for dataset of input and target output pairs. Most AutoML algorithms are based on RL or evolutionary algorithms. Examples of AutoML algorithms include AdaNet[1400,1401], Auto-DeepLab[1402], AutoGAN[1403], Auto-Keras[1404], auto-sklearn[1405], DARTS+[1406], EvoCNN[261], Ludwig[1407], MENNDL[1408,1409], NASBOT[1410], XNAS[1411], and others[1412–1416]. AutoML is becoming increasingly popular as it can achieve higher performance than human developers[1053] and enables human developer time to be traded for potentially cheaper computer time. Nevertheless, AutoML is currently limited to established ANN architectures and learning policies. Following, we recommend that researchers either focus on novel ANN architectures and learning policies or developing ANNs for novel applications.

6.4 Initialization

How ANN trainable parameters are initialized[525,1417] is related to model capacity[1418]. Further, initializing parameters with values that are too small or large can cause slow learning or divergence[525]. Careful initialization can also prevent training by gradient descent being destabilized by vanishing or exploding gradients[525,1185,1186], or high variance of length scales across layers[525]. Finally, careful initialization can enable momentum to accelerate convergence and improve performance[1273]. Most trainable parameters are multiplicative weights or additive biases. Initializing parameters with constant values would result in every parameter in a layer receiving the same updates by gradient descent, reducing model capacity. Thus, weights are often randomly initialized. Following, biases are often initialized with constant values due to symmetry breaking by the weights.

Consider the projection of n_{in} inputs, $\mathbf{x}^{\text{input}} = \{x_1^{\text{input}}, ..., x_{n_{\text{in}}}^{\text{input}}\}$, to n_{out} outputs, $\mathbf{x}^{\text{output}} = \{x_1^{\text{output}}, ..., x_{n_{\text{out}}}^{\text{output}}\}$, by an $n_{\text{in}} \times n_{\text{out}}$ weight matrix, $\mathbf{w}$. The expected variance of an output element is[1417]

$$\text{Var}(\mathbf{x}^{\text{output}}) = n_{\text{in}}\text{E}(\mathbf{x}^{\text{input}})^2\text{Var}(\mathbf{w}) + n_{\text{in}}\text{E}(\mathbf{w})^2\text{Var}(\mathbf{x}^{\text{input}}) + n_{\text{in}}\text{Var}(\mathbf{w})\text{Var}(\mathbf{x}^{\text{input}}), \tag{93}$$

where $\text{E}(\mathbf{x})$ and $\text{Var}(\mathbf{x})$ denote the expected mean and variance of elements of $\mathbf{x}$, respectively. For similar length scales across layers, $\text{Var}(\mathbf{x}^{\text{output}})$ should be constant. Initially, similar variances can be achieved by normalizing ANN inputs to have zero mean, so that $\text{E}(\mathbf{x}^{\text{input}}) = 0$, and initializing weights so that $\text{E}(\mathbf{w}) = 0$ and $\text{Var}(\mathbf{w}) = 1/n_{\text{in}}$. However, parameters can shift during training, destabilizing learning. To compensate for parameter shift, popular normalization layers like batch normalization often impose $\text{E}(\mathbf{x}^{\text{input}}) = 0$ and $\text{Var}(\mathbf{x}^{\text{input}}) = 1$, relaxing need for $\text{E}(\mathbf{x}^{\text{input}}) = 0$ or $\text{E}(\mathbf{w}) = 0$. Nevertheless, training will still be sensitive to the length scale of trainable parameters.

There are a variety of popular weight initializers that adapt weights to ANN architecture. One of the oldest methods is LeCun initialization[917,927], where weights are initialized with variance,

$$\text{Var}(\mathbf{w}) = \frac{1}{n_{\text{in}}}, \tag{94}$$

which is argued to produce outputs with similar length scales in the previous paragraph. However, a similar argument can be made for initializing with $\text{Var}(\mathbf{w}) = 1/n_{\text{out}}$ to produce similar gradients at each layer during the backwards pass[1417]. As a compromise, Xavier initialization[1419] computes an average,

$$\text{Var}(\mathbf{w}) = \frac{2}{n_{\text{in}} + n_{\text{out}}}. \tag{95}$$

However, adjusting weights for n_{out} is not necessary for adaptive optimizers like ADAM, which divide gradients by their lengths scales, unless gradients will vanish or explode. Finally, He initialization[20] doubles the variance of weights to

$$\text{Var}(\mathbf{w}) = \frac{2}{n_{\text{in}}}, \tag{96}$$

and is often used in ReLU networks to compensate for activation functions halving variances of their outputs[20,1417,1420]. Most trainable parameters are initialized from either a zero-centred Gaussian or uniform distribution. For convenience, the limits of such a uniform distribution are $\pm(3\mathrm{Var}(\mathbf{w}))^{1/2}$. Uniform initialization can outperform Gaussian initialization in DNNs due to Gaussian outliers harming learning[1417]. However, issues can be avoided by truncating Gaussian initialization, often to two standard deviations, and rescaling to its original variance.

Some initializers are mainly used for RNNs. For example, orthogonal initialization[1421] often improves RNN training[1422] by reducing susceptibility to vanishing and exploding gradients. Similarly, identity initialization[1423,1424] can help RNNs to learn long-term dependencies. In most ANNs, biases are initialized with zeros. However, the forget gates of LSTMs are often initialized with ones to decrease forgetting at the start of training[1188]. Finally, the start states of most RNNs are initialized with zeros or other constants. However, random multivariate or trainable variable start states can improve performance[1425].

There are a variety of alternatives to initialization from random multivariates. Weight normalized[985] ANNs are a popular example of data-dependent initialization, where randomly initialized weight magnitudes and biases are chosen to counteract variances and means of an initial batch of data. Similarly, layer-sequential unit-variance (LSUV) initialization[1426] consists of orthogonal initialization followed by adjusting the magnitudes of weights to counteract variances of an initial batch of data. Other approaches standardize the norms of backpropagated gradient. For example, random walk initialization[1427] (RWI) finds scales for weights to prevent vanishing or exploding gradients in deep FNNs, albeit with varied success[1426]. Alternatively, MetaInit[1428] scales the magnitudes of randomly initialized weights to minimize changes in backpropagated gradients per iteration of gradient descent.

6.5 Regularization

There are a variety of regularization mechanisms[1429–1432] that modify learning algorithms to improve ANN performance. One of the most popular is LX regularization, which decays weights by adding a loss,

$$L_X = \lambda_X \sum_i \frac{|\theta_i|^X}{X}, \tag{97}$$

weighted by λ_X to each trainable variable, θ_i. L2 regularization[1433–1435] is preferred[1436] for most DNN optimization as subtraction of its gradient, $\partial_{\theta_i} L_2 = \lambda_2 \theta_i$, is equivalent to computationally-efficient multiplicative weight decay. Nevertheless, L1 regularization is better at inducing model sparsity[1437] than L2 regularization, and L1 regularization achieves higher performance in some applications[1438]. Higher performance can also be achieved by adding both L1 and L2 regularization in elastic nets[1439]. LX regularization is most effective at the start of training and becomes less important near convergence[1433]. Finally, L1 and L2 regularization are closely related to lasso[1440] and ridge[1441] regularization, respectively, whereby trainable parameters are adjusted to limit L_1 and L_2 penalties.

Gradient clipping[1313,1442–1444] accelerates learning by limiting large gradients, and is most commonly applied to RNNs. A simple approach is to clip gradient magnitudes to a threshold hyperparameter. However, it is more common to scale gradients, $\mathbf{g}_i$, at layer i if their norm is above a threshold, u, so that[1185,1443]

$$\mathbf{g}_i \leftarrow \begin{cases} \mathbf{g}_i, & \text{if } ||\mathbf{g}_i||_n \leq u \\ \frac{u}{||\mathbf{g}_i||_n}\mathbf{g}_i, & \text{if } ||\mathbf{g}_i||_n > u \end{cases} \tag{98}$$

where $n = 2$ is often chosen to minimize computation. Similarly, gradients can be clipped if they are above a global norm,

$$g_{\text{norm}} = \left(\sum_{i=1}^{L} ||\mathbf{g}_i||_n^n, \right)^{1/n} \tag{99}$$

computed with gradients at L layers. Scaling gradient norms is often preferable to clipping to a threshold as scaling is akin to adapting layer learning rates and does not affect the directions of gradients. Thresholds for gradient clipping are often set based on average norms of backpropagated gradients during preliminary training[1445]. However, thresholds can also be set automatically and adaptively[1312,1313]. In addition, adaptive gradient clipping algorithms can skip training iterations if gradient norms are anomalously high[1446], which often indicates an imminent gradient explosion.

Dropout[1447–1451] often reduces overfitting by only using a fraction, p_i, of layer i outputs during training, and multiplying all outputs by p_i for inference. However, dropout often increases training time, can be sensitive to p_i, and sometimes lowers performance[1452]. Improvements to dropout at the structural level, such as applying it to convolutional channels, paths, and layers, rather than random output elements, can improve performance[1453]. For example, DropBlock[1454] improves performance by dropping contiguous regions of feature maps to prevent dropout being trivially circumvented by using spatially correlated neighbouring outputs. Similarly, PatchUp[1455] swaps or mixes contiguous regions with regions for another sample. Dropout is

often outperformed by Shakeout[1456,1457], a modification of dropout that randomly enhances or reverses contributions of outputs to the next layer.

Noise often enhances ANN training by decreasing susceptibility to spurious local minima[1458]. Adding noise to trainable parameters can improve generalization[1459,1460], or exploration for RL[1381]. Parameter noise is usually additive as it does not change an objective function being learned, whereas multiplicative noise can change the objective[1461]. In addition, noise can be added to inputs[1230,1462], hidden layers[1134,1463], generated outputs[1464] or target outputs[971,1465]. However, adding noise to signals does not always improve performance[1194]. Finally, modifying usual gradient noise[1466] by adding noise to gradients can improve performance[1467]. Typically, additive noise is annealed throughout training, so that that final training is with a noiseless model that will be used for inference.

There are a variety of regularization mechanism that exploit extra training data. A simple approach is to create extra training examples by data augmentation[1468–1470]. Extra training data can also be curated, or simulated for training by domain adaption[1153–1155]. Alternatively, semi-supervised learning[1471–1476] can generate target outputs for dataset of unpaired inputs to augment training with a dataset of paired inputs and target outputs. Finally, multitask learning[1477–1481] can improve performance by introducing additional loss functions. For instance, by adding an auxiliary classifier to predict image labels from features generated by intermediate DNN layers[1482–1485]. Losses are often manually balanced; however, their gradients can also be balanced automatically and adaptively[1143,1144].

6.6 Data Pipeline

A data pipeline prepares data to be input to an ANN. Most data parallelize data pipelines across multiple CPU cores[1486]. Small datasets can be stored in RAM to decrease data access times, whereas large dataset elements are often loaded from files. Loaded data can then be preprocessed and augmented[1468,1469,1487–1489]. For electron micrographs, preprocessing often includes replacing non-finite elements, such as Nan and inf, with finite values; linearly transforming intensities to a common range, such as $[-1, 1]$ or zero mean and unit variance; and performing a random combination of flips and 90° to augment data by a factor of eight[66,192,193,220,337]. Preprocessed examples can then be combined into batches. Typically, multiple batches that are ready to be input are prefetched and stored in RAM to avoid delays due to fluctuating CPU performance.

To efficiently utilize data, training datasets are often reiterated over for multiple training epochs. Usually, training datasets are reiterated over about 10^2 times. Increasing epochs can maximize utilization of potentially expensive training data; however, increasing epochs can lower performance due to overfitting[1490,1491] or be too computationally expensive[527]. Naively, batches of data can be randomly sampled with replacement during training by gradient descent. However, convergence can be accelerated by reinitializing a training dataset at the start of each training epoch and randomly sampling data without replacement[1492–1496]. Most modern DLFs, such as TensorFlow, provide efficient and easy-to-use functions to control data sampling[1497].

6.7 Model Evaluation

There are a variety of methods for ANN performance evaluation[526]. However, most ANNs are evaluated by 1-fold validation, where a dataset is partitioned into training, validation, and test sets. After ANN optimization with a training set, ability to generalize is measured with a validation set. Multiple validations may be performed for training with early stopping[1490,1491] or ANN learning policy and architecture selection, so final performance is often measured with a test set to avoid overfitting to the validation set. Most researchers favor using single training, validation, and test sets to simplify standardization of performance benchmarks[220]. However, multiple-fold validation[526] or multiple validation sets[1498] can improve performance characterization. Alternatively, models can be bootstrap aggregated[1499] (bagged) from multiple models trained on different subsets of training data. Bagging is usually applied to random forests[1500–1502] or other lightweight models, and enables model uncertainly to be gauged from the variance of model outputs.

For small datasets, model performance is often sensitive to split of data between training and validation sets[1503]. Increasing training set size usually increases model accuracy, whereas increasing validation set size decreases performance uncertainty. Indeed, a scaling law can be used to estimate an optimal tradeoff[1504] between training and validation set sizes. However, most experimenters follow a Pareto[1505] splitting heuristic. For example, we often use a 75:15:10 training-validation-test split[220]. Heuristic splitting is justified for ANN training with large datasets insofar that sensitivity to splitting ratios decreases with increasing dataset size[2].

6.8 Deployment

If an ANN is deployed[1506–1508] on multiple different devices, such as various electron microscopes, a separate model can be trained for each device[391], Alternatively, a single model can be trained and specialized for different devices to decrease training requirements[1509]. In addition, ANNs can remotely service requests from cloud containers[1510–1512]. Integration of multiple ANNs can be complicated by different servers for different DLFs supporting different backends; however, unified interfaces are available. For example, GraphPipe[1513] provides simple, efficient reference model servers for Tensorflow, Caffe2, and ONNX;

a minimalist machine learning transport specification based on FlatBuffers[1514]; and efficient client implementations in Go, Python, and Java. In 2020, most ANNs developed researchers were not deployed. However, we anticipate that deployment will become a more prominent consideration as the role of deep learning in electron microscopy matures.

Most ANNs are optimized for inference by minimizing parameters and operations from training time, like MobileNets[1070]. However, less essential operations can also be pruned after training[1515,1516]. Another approach is quantization, where ANN bit depths are decreased, often to efficient integer instructions, to increase inference throughput[1517,1518]. Quantization often decreases performance; however, the amount of quantization can be adapted to ANN components to optimize performance-throughput tradeoffs[1519]. Alternatively, training can be modified to minimize the impact of quantization on performance[1520–1522]. Another approach is to specialize bit manipulation for deep learning. For example, signed brain floating point (bfloat16) often improves accuracy on TPUs by using an 8 bit mantissa and 7 bit exponent, rather than a usual 5 bit mantissa and 10 bit exponent[1523]. Finally, ANNs can be adaptively selected from a set of ANNs based on available resources to balance tradeoff of performance and inference time[1524], similar to image optimization for web applications[1525,1526].

Figure 16. Inputs that maximally activate channels in GoogLeNet[1052] after training on ImageNet[67]. Neurons in layers near the start have small receptive fields and discern local features. Middle layers discern semantics recognisable by humans, such as wheels and dogs. Finally, layers at the end of the DNN, near its logits, discern combinations of semantics that are useful for labelling. This figure is adapted with permission[1527] under a Creative Commons 4.0[69] license.

6.9 Interpretation

We observe that some electron microscopists are apprehensive about working with ANNs due to a lack of interpretability, irrespective of rigorous ANN validation. We try to address uncertainty by providing loss visualizations in some of our electron microscopy papers[66,192,193]. However, there are a variety of popular approaches to explainable artificial intelligence[1528–1534] (XAI). One of the most popular approaches to XAI is saliency, where gradients of outputs w.r.t. inputs correlate with their importance. Saliency is often computed by gradient backpropagation[1535–1537]. For example, with Grad-CAM[1538] or its variants[1539–1542]. Alternatively, saliency can be predicted by ANNs[1030,1543–1545] or a variety of methods inspired by Grad-CAM[1546–1548]. Saliency maps can be exploited to select features[1549].

There are a variety of other approaches to XAI. For example, feature visualization via optimization[1527,1550–1553] can find inputs that maximally activate neurons, as shown in fig. 16. Another approach is to cluster features, e.g. by tSNE[1554,1555] with the Barnes-Hut algorithm[1556,1557], and examine corresponding clustering of inputs or outputs[220]. More simply, researchers can view raw features and gradients during forward and backward passes of gradient descent, respectively. For example, CNN explainer[1558,1559] is an interactive visualization tool designed for non-experts to learn and experiment with CNNs. Similarly, GAN Lab[1560] is an interactive visualization tool for GANs.

7 Discussion

We introduced a variety of electron microscopy applications in section 1 that have been enabled or enhanced by deep learning. However, the greatest benefit of deep learning in electron microscopy may be general-purpose tools that enable researchers to be more effective. Search engines based on deep learning are almost essential to navigate an ever-increasing number of scientific publications[685]. Further, machine learning can enhance communication by filtering spam and phishing attacks[1561–1563], and by summarizing[1564–1566] and classifying[1031,1567–1569] scientific documents. In addition, machine learning can be applied to education to automate and standardize scoring[1570–1573], detect plagiarism[1574–1576], and identify at-risk students[1577].

Creative applications of deep learning[1578,1579] include making new art by style transfer[1020,1149,1580,1581], composing music[1582–1584], and storytelling[1585,1586]. Similar DNNs can assist programmers[1587,1588]. For example, by predictive source code completion[1589–1594], and by generating source code to map inputs to target outputs[1595] or from labels describing desired source code[1596]. Text generating DNNs can also help write scientific papers. For example, by drafting scientific passages[1597] or drafting part of a paper from a list of references[1598]. Papers generated by early prototypes for automatic scientific paper generators, such as SciGen[1599], are realistic insofar that they have been accepted by scientific venues.

An emerging application of deep learning is mining scientific resources to make new scientific discoveries[1600]. Artificial agents are able to effectively distil latent scientific knowledge as they can parallelize examination of huge amounts of data, whereas information access by humans[1601–1603] is limited by human cognition[1604]. High bandwidth bi-directional brain-machine interfaces are being developed to overcome limitations of human cognition[1605]; however, they are in the early stages of development and we expect that they will depend on substantial advances in machine learning to enhance control of cognition. Eventually, we expect that ANNs will be used as scientific oracles, where researchers who do not rely on their services will no longer be able to compete. For example, an ANN trained on a large corpus of scientific literature predicted multiple advances in materials science before they were reported[1606]. ANNs are already used for financial asset management[1607,1608] and recruiting[1609–1612], so we anticipate that artificial scientific oracle consultation will become an important part of scientific grant[1613,1614] reviews.

A limitation of deep learning is that it can introduce new issues. For example, DNNs are often susceptible to adversarial attacks[1615–1619], where small perturbations to inputs cause large errors. Nevertheless, training can be modified to improve robustness to adversarial attacks[1620–1623]. Another potential issue is architecture-specific systematic errors. For example, CNNs often exhibit structured systematic error variation[66,192,193,1068,1069,1624], including higher errors nearer output edges[66,192,193]. However, structured systematic error variation can be minimized by GANs incentifying the generation of realistic outputs[192]. Finally, ANNs can be difficult to use as they often require downloading code with undocumented dependencies, downloading a pretrained model, and may require hardware accelerators. These issues can be avoided by serving ANNs from cloud containers. However, it may not be practical for academics to acquire funding to cover cloud service costs.

Perhaps the most important aspect of deep learning in electron microscopy is that it presents new challenges that can lead to advances in machine learning. Simple benchmarks like CIFAR-10[550,551] and MNIST[552] have been solved. Following, more difficult benchmark like Fashion-MNIST[1625] have been introduced. However, they only partially address issues with solved datasets as they do not present fundamentally new challenges. In contrast, we believe that new problems often invite new solutions. For example, we developed adaptive learning rate clipping[251] to stabilize training of DNNs for partial scanning transmission electron microscopy[192]. The challenge was that we wanted to train a large model for high-resolution images; however, training was unstable if we used small batches needed to fit it in GPU memory. Similar challenges abound and can lead to advances in both machine learning and electron microscopy.

Acknowledgements

J.M.E. acknowledges EPSRC grant EP/N035437/1 and EPSRC Studentship 1917382.

8 Competing Interests

The author declares no competing interests.

References

1. Leiserson, C. E. *et al.* There's Plenty of Room at the Top: What Will Drive Computer Performance After Moore's Law? *Science* **368** (2020).
2. Sun, C., Shrivastava, A., Singh, S. & Gupta, A. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision*, 843–852 (2017).
3. Sengupta, S. *et al.* A Review of Deep Learning with Special Emphasis on Architectures, Applications and Recent Trends. *Knowledge-Based Syst.* 105596 (2020).
4. Shrestha, A. & Mahmood, A. Review of Deep Learning Algorithms and Architectures. *IEEE Access* **7**, 53040–53065 (2019).
5. Dargan, S., Kumar, M., Ayyagari, M. R. & Kumar, G. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Arch. Comput. Methods Eng.* 1–22 (2019).
6. Alom, M. Z. *et al.* A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics* **8**, 292 (2019).
7. Zhang, Q., Yang, L. T., Chen, Z. & Li, P. A Survey on Deep Learning for Big Data. *Inf. Fusion* **42**, 146–157 (2018).
8. Hatcher, W. G. & Yu, W. A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. *IEEE Access* **6**, 24411–24432 (2018).
9. LeCun, Y., Bengio, Y. & Hinton, G. Deep Learning. *Nature* **521**, 436–444 (2015).
10. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Networks* **61**, 85–117 (2015).
11. von Lilienfeld, O. A. Introducing Machine Learning: Science and Technology. *Mach. Learn. Sci. Technol.* **1**, 010201 (2020).
12. Carleo, G. *et al.* Machine Learning and the Physical Sciences. *Rev. Mod. Phys.* **91**, 045002 (2019).
13. Wei, J. *et al.* Machine Learning in Materials Science. *InfoMat* **1**, 338–358 (2019).
14. Barbastathis, G., Ozcan, A. & Situ, G. On the Use of Deep Learning for Computational Imaging. *Optica* **6**, 921–943 (2019).
15. Schleder, G. R., Padilha, A. C., Acosta, C. M., Costa, M. & Fazzio, A. From DFT to Machine Learning: Recent Approaches to Materials Science – A Review. *J. Physics: Mater.* **2**, 032001 (2019).
16. Sejnowski, T. J. *The Deep Learning Revolution* (MIT Press, 2018).
17. Alom, M. Z. *et al.* The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *arXiv preprint arXiv:1803.01164* (2018).
18. Wang, Y. & Kosinski, M. Deep Neural Networks are More Accurate than Humans at Detecting Sexual Orientation from Facial Images. *J. Pers. Soc. Psychol.* **114**, 246 (2018).
19. Kheradpisheh, S. R., Ghodrati, M., Ganjtabesh, M. & Masquelier, T. Deep Networks can Resemble Human Feed-Forward Vision in Invariant Object Recognition. *Sci. Reports* **6**, 32672 (2016).
20. He, K., Zhang, X., Ren, S. & Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 1026–1034 (2015).
21. Lu, C. & Tang, X. Surpassing Human-Level Face Verification Performance on LFW with GaussianFace. In *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).
22. Vinyals, O. *et al.* AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. Online: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (2019).
23. Firoiu, V., Whitney, W. F. & Tenenbaum, J. B. Beating the World's Best at Super Smash Bros. with Deep Reinforcement Learning. *arXiv preprint arXiv:1702.06230* (2017).
24. Lample, G. & Chaplot, D. S. Playing FPS Games with Deep Reinforcement Learning. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).
25. Silver, D. *et al.* Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* **529**, 484–489 (2016).
26. Mnih, V. *et al.* Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602* (2013).
27. Tesauro, G. Programming Backgammon Using Self-Teaching Neural Nets. *Artif. Intell.* **134**, 181–199 (2002).

28. Han, S. S. *et al.* Deep Neural Networks Show an Equivalent and Often Superior Performance to Dermatologists in Onychomycosis Diagnosis: Automatic Construction of Onychomycosis Datasets by Region-Based Convolutional Deep Neural Network. *PLOS ONE* **13**, e0191493 (2018).

29. Wang, D., Khosla, A., Gargeya, R., Irshad, H. & Beck, A. H. Deep Learning for Identifying Metastatic Breast Cancer. *arXiv preprint arXiv:1606.05718* (2016).

30. Santoro, A. *et al.* A Simple Neural Network Module for Relational Reasoning. In *Advances in Neural Information Processing Systems*, 4967–4976 (2017).

31. Xiong, W. *et al.* Achieving Human Parity in Conversational Speech Recognition. *arXiv preprint arXiv:1610.05256* (2016).

32. Weng, C., Yu, D., Seltzer, M. L. & Droppo, J. Single-Channel Mixed Speech Recognition Using Deep Neural Networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5632–5636 (IEEE, 2014).

33. Lee, K., Zung, J., Li, P., Jain, V. & Seung, H. S. Superhuman Accuracy on the SNEMI3D Connectomics Challenge. *arXiv preprint arXiv:1706.00120* (2017).

34. Weyand, T., Kostrikov, I. & Philbin, J. Planet-Photo Geolocation with Convolutional Neural Networks. In *European Conference on Computer Vision*, 37–55 (Springer, 2016).

35. Kidger, P. & Lyons, T. Universal Approximation with Deep Narrow Networks. *arXiv preprint arXiv:1905.08539* (2019).

36. Lin, H. & Jegelka, S. ResNet with One-Neuron Hidden Layers is a Universal Approximator. In *Advances in Neural Information Processing Systems*, 6169–6178 (2018).

37. Hanin, B. & Sellke, M. Approximating Continuous Functions by ReLU Nets of Minimal Width. *arXiv preprint arXiv:1710.11278* (2017).

38. Lu, Z., Pu, H., Wang, F., Hu, Z. & Wang, L. The Expressive Power of Neural Networks: A View from the Width. In *Advances in Neural Information Processing Systems*, 6231–6239 (2017).

39. Pinkus, A. Approximation Theory of the MLP Model in Neural Networks. *Acta Numer.* **8**, 143–195 (1999).

40. Leshno, M., Lin, V. Y., Pinkus, A. & Schocken, S. Multilayer Feedforward Networks with a Nonpolynomial Activation Function can Approximate any Function. *Neural Networks* **6**, 861–867 (1993).

41. Hornik, K. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks* **4**, 251–257 (1991).

42. Hornik, K., Stinchcombe, M. & White, H. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks* **2**, 359–366 (1989).

43. Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. *Math. Control. Signals Syst.* **2**, 303–314 (1989).

44. Johnson, J. Deep, Skinny Neural Networks are not Universal Approximators. *arXiv preprint arXiv:1810.00393* (2018).

45. Lin, H. W., Tegmark, M. & Rolnick, D. Why Does Deep and Cheap Learning Work so Well? *J. Stat. Phys.* **168**, 1223–1247 (2017).

46. Raghu, M., Poole, B., Kleinberg, J., Ganguli, S. & Sohl-Dickstein, J. On the Expressive Power of Deep Neural Networks. In *International Conference on Machine Learning*, 2847–2854 (2017).

47. Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J. & Ganguli, S. Exponential Expressivity in Deep Neural Networks Through Transient Chaos. In *Advances in Neural Information Processing Systems*, 3360–3368 (2016).

48. Cao, Y. & Gu, Q. Generalization Error Bounds of Gradient Descent for Learning Over-Parameterized Deep ReLU Networks. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 3349–3356 (2020).

49. Geiger, M. *et al.* Scaling Description of Generalization with Number of Parameters in Deep Learning. *J. Stat. Mech. Theory Exp.* **2020**, 023401 (2020).

50. Dziugaite, G. K. *Revisiting Generalization for Deep Learning: PAC-Bayes, Flat Minima, and Generative Models*. Ph.D. thesis, University of Cambridge (2020).

51. Cao, Y. & Gu, Q. Generalization Bounds of Stochastic Gradient Descent for Wide and Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 10836–10846 (2019).

52. Xu, Z. J. Understanding Training and Generalization in Deep Learning by Fourier Analysis. *arXiv preprint arXiv:1808.04295* (2018).

53. Neyshabur, B., Bhojanapalli, S., McAllester, D. & Srebro, N. Exploring Generalization in Deep Learning. In *Advances in Neural Information Processing systems*, 5947–5956 (2017).

54. Wu, L., Zhu, Z. *et al.* Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes. *arXiv preprint arXiv:1706.10239* (2017).

55. Kawaguchi, K., Kaelbling, L. P. & Bengio, Y. Generalization in Deep Learning. *arXiv preprint arXiv:1710.05468* (2017).

56. Iten, R., Metger, T., Wilming, H., Del Rio, L. & Renner, R. Discovering Physical Concepts with Neural Networks. *Phys. Rev. Lett.* **124**, 010508 (2020).

57. Wu, T. & Tegmark, M. Toward an Artificial Intelligence Physicist for Unsupervised Learning. *Phys. Rev. E* **100**, 033311 (2019).

58. Chen, Y., Xie, Y., Song, L., Chen, F. & Tang, T. A Survey of Accelerator Architectures for Deep Neural Networks. *Engineering* **6**, 264–274 (2020).

59. Garrido, M., Qureshi, F., Takala, J. & Gustafsson, O. Hardware Architectures for the Fast Fourier Transform. In *Handbook of Signal Processing Systems*, 613–647 (Springer, 2019).

60. Velik, R. Discrete Fourier Transform Computation Using Neural Networks. In *2008 International Conference on Computational Intelligence and Security*, 120–123 (IEEE, 2008).

61. Moreland, K. & Angel, E. The FFT on a GPU. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, 112–119 (Eurographics Association, 2003).

62. Breen, P. G., Foley, C. N., Boekholt, T. & Zwart, S. P. Newton Versus the Machine: Solving the Chaotic Three-Body Problem Using Deep Neural Networks. *Mon. Notices Royal Astron. Soc.* **494** (2020).

63. Ryczko, K., Strubbe, D. A. & Tamblyn, I. Deep Learning and Density-Functional Theory. *Phys. Rev. A* **100**, 022512 (2019).

64. Sinitskiy, A. V. & Pande, V. S. Deep Neural Network Computes Electron Densities and Energies of a Large Set of Organic Molecules Faster than Density Functional Theory (DFT). *arXiv preprint arXiv:1809.02723* (2018).

65. Zhang, G. *et al.* Fast Phase Retrieval in Off-Axis Digital Holographic Microscopy Through Deep Learning. *Opt. Express* **26**, 19388–19405 (2018).

66. Ede, J. M. & Beanland, R. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *Ultramicroscopy* **202**, 18–25 (2019).

67. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 1097–1105 (2012).

68. Ede, J. M. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *arXiv preprint arXiv:1807.11234* (2018).

69. Creative Commons Attribution 4.0 International (CC BY 4.0). Online: https://creativecommons.org/licenses/by/4.0 (2020).

70. Liu, B. & Liu, J. Overview of Image Denoising Based on Deep Learning. In *Journal of Physics: Conference Series*, vol. 1176, 022010 (IOP Publishing, 2019).

71. Tian, C. *et al.* Deep Learning on Image Denoising: An Overview. *arXiv preprint arXiv:1912.13171* (2019).

72. Yoon, D., Lim, H. S., Jung, K., Kim, T. Y. & Lee, S. Deep Learning-Based Electrocardiogram Signal Noise Detection and Screening Model. *Healthc. Informatics Res.* **25**, 201–211 (2019).

73. Antczak, K. Deep Recurrent Neural Networks for ECG Signal Denoising. *arXiv preprint arXiv:1807.11551* (2018).

74. Bai, T., Nguyen, D., Wang, B. & Jiang, S. Probabilistic Self-Learning Framework for Low-Dose CT Denoising. *arXiv preprint arXiv:2006.00327* (2020).

75. Jifara, W., Jiang, F., Rho, S., Cheng, M. & Liu, S. Medical Image Denoising Using Convolutional Neural Network: A Residual Learning Approach. *The J. Supercomput.* **75**, 704–718 (2019).

76. Feng, D., Wu, W., Li, H. & Li, Q. Speckle Noise Removal in Ultrasound Images Using a Deep Convolutional Neural Network and a Specially Designed Loss Function. In *International Workshop on Multiscale Multimodal Medical Imaging*, 85–92 (Springer, 2019).

77. de Haan, K., Rivenson, Y., Wu, Y. & Ozcan, A. Deep-Learning-Based Image Reconstruction and Enhancement in Optical Microscopy. *Proc. IEEE* **108**, 30–50 (2019).

78. Manifold, B., Thomas, E., Francis, A. T., Hill, A. H. & Fu, D. Denoising of Stimulated Raman Scattering Microscopy Images via Deep Learning. *Biomed. Opt. Express* **10**, 3860–3874 (2019).

79. Devalla, S. K. *et al.* A Deep Learning Approach to Denoise Optical Coherence Tomography Images of the Optic Nerve Head. *Sci. Reports* **9**, 1–13 (2019).

80. Choi, G. *et al.* Cycle-Consistent Deep Learning Approach to Coherent Noise Reduction in Optical Diffraction tomography. *Opt. Express* **27**, 4927–4943 (2019).

81. Azarang, A. & Kehtarnavaz, N. A Review of Multi-Objective Deep Learning Speech Denoising Methods. *Speech Commun.* (2020).

82. Choi, H.-S., Heo, H., Lee, J. H. & Lee, K. Phase-Aware Single-Stage Speech Denoising and Dereverberation with U-Net. *arXiv preprint arXiv:2006.00687* (2020).

83. Alamdari, N., Azarang, A. & Kehtarnavaz, N. Self-Supervised Deep Learning-Based Speech Denoising. *arXiv* arXiv–1904 (2019).

84. Han, K. *et al.* Learning Spectral Mapping for Speech Dereverberation and Denoising. *IEEE/ACM Transactions on Audio, Speech, Lang. Process.* **23**, 982–992 (2015).

85. Goyal, B., Dogra, A., Agrawal, S., Sohi, B. & Sharma, A. Image Denoising Review: From Classical to State-of-the-Art Approaches. *Inf. Fusion* **55**, 220–244 (2020).

86. Girdher, A., Goyal, B., Dogra, A., Dhindsa, A. & Agrawal, S. Image Denoising: Issues and Challenges. *Available at SSRN 3446627* (2019).

87. Fan, L., Zhang, F., Fan, H. & Zhang, C. Brief Review of Image Denoising Techniques. *Vis. Comput. for Ind. Biomed. Art* **2**, 7 (2019).

88. Gedraite, E. S. & Hadad, M. Investigation on the Effect of a Gaussian Blur in Image Filtering and Segmentation. In *Proceedings ELMAR*, 393–396 (IEEE, 2011).

89. Deng, G. & Cahill, L. An Adaptive Gaussian Filter for Noise Reduction and Edge Detection. In *1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference*, 1615–1619 (IEEE, 1993).

90. Chang, H.-H., Lin, Y.-J. & Zhuang, A. H. An Automatic Parameter Decision System of Bilateral Filtering with GPU-Based Acceleration for Brain MR Images. *J. Digit. Imaging* **32**, 148–161 (2019).

91. Chaudhury, K. N. & Rithwik, K. Image Denoising Using Optimally Weighted Bilateral Filters: A Sure and Fast Approach. In *IEEE International Conference on Image Processing*, 108–112 (IEEE, 2015).

92. Anantrasirichai, N. *et al.* Adaptive-Weighted Bilateral Filtering and Other Pre-Processing Techniques for Optical Coherence Tomography. *Comput. Med. Imaging Graph.* **38**, 526–539 (2014).

93. Tomasi, C. & Manduchi, R. Bilateral Filtering for Gray and Color Images. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, 839–846 (IEEE, 1998).

94. Budhiraja, S., Goyal, B., Dogra, A., Agrawal, S. *et al.* An Efficient Image Denoising Scheme for Higher Noise Levels Using Spatial Domain Filters. *Biomed. Pharmacol. J.* **11**, 625–634 (2018).

95. Nair, R. R., David, E. & Rajagopal, S. A Robust Anisotropic Diffusion Filter with Low Arithmetic Complexity for Images. *EURASIP J. on Image Video Process.* **2019**, 48 (2019).

96. Perona, P. & Malik, J. Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis Mach. Intell.* **12**, 629–639 (1990).

97. Wang, Z. & Zhang, D. Progressive Switching Median Filter for the Removal of Impulse Noise from Highly Corrupted Images. *IEEE Transactions on Circuits Syst. II: Analog. Digit. Signal Process.* **46**, 78–80 (1999).

98. Yang, R., Yin, L., Gabbouj, M., Astola, J. & Neuvo, Y. Optimal Weighted Median Filtering Under Structural Constraints. *IEEE Transactions on Signal Process.* **43**, 591–604 (1995).

99. Kodi Ramanah, D., Lavaux, G. & Wandelt, B. D. Wiener Filter Reloaded: Fast Signal Reconstruction Without Preconditioning. *Mon. Notices Royal Astron. Soc.* **468**, 1782–1793 (2017).

100. Elsner, F. & Wandelt, B. D. Efficient Wiener Filtering Without Preconditioning. *Astron. & Astrophys.* **549**, A111 (2013).

101. Robinson, E. A. & Treitel, S. Principles of Digital Wiener Filtering. *Geophys. Prospect.* **15**, 311–332 (1967).

102. Bayer, F. M., Kozakevicius, A. J. & Cintra, R. J. An Iterative Wavelet Threshold for Signal Denoising. *Signal Process.* **162**, 10–20 (2019).

103. Mohideen, S. K., Perumal, S. A. & Sathik, M. M. Image De-Noising Using Discrete Wavelet Transform. *Int. J. Comput. Sci. Netw. Secur.* **8**, 213–216 (2008).

104. Luisier, F., Blu, T. & Unser, M. A New SURE Approach to Image Denoising: Interscale Orthonormal Wavelet Thresholding. *IEEE Transactions on Image Process.* **16**, 593–606 (2007).

105. Jansen, M. & Bultheel, A. Empirical Bayes Approach to Improve Wavelet Thresholding for Image Noise Reduction. *J. Am. Stat. Assoc.* **96**, 629–639 (2001).

106. Chang, S. G., Yu, B. & Vetterli, M. Adaptive Wavelet Thresholding for Image Denoising and Compression. *IEEE Transactions on Image Process.* **9**, 1532–1546 (2000).

107. Donoho, D. L. & Johnstone, J. M. Ideal Spatial Adaptation by Wavelet Shrinkage. *Biometrika* **81**, 425–455 (1994).

108. Ma, J. & Plonka, G. The Curvelet Transform. *IEEE Signal Process. Mag.* **27**, 118–133 (2010).

109. Starck, J.-L., Candès, E. J. & Donoho, D. L. The Curvelet Transform for Image Denoising. *IEEE Transactions on Image Process.* **11**, 670–684 (2002).

110. Ahmed, S. S. *et al.* Nonparametric Denoising Methods Based on Contourlet Transform with Sharp Frequency Localization: Application to Low Exposure Time Electron Microscopy Images. *Entropy* **17**, 3461–3478 (2015).

111. Do, M. N. & Vetterli, M. The Contourlet Transform: An Efficient Directional Multiresolution Image Representation. *IEEE Transactions on Image Process.* **14**, 2091–2106 (2005).

112. Diwakar, M. & Kumar, P. Wavelet Packet Based CT Image Denoising Using Bilateral Method and Bayes Shrinkage Rule. In *Handbook of Multimedia Information Security: Techniques and Applications*, 501–511 (Springer, 2019).

113. Thakur, K., Damodare, O. & Sapkal, A. Hybrid Method for Medical Image Denoising Using Shearlet Transform and Bilateral Filter. In *2015 International Conference on Information Processing (ICIP)*, 220–224 (IEEE, 2015).

114. Nagu, M. & Shanker, N. V. Image De-Noising by Using Median Filter and Weiner Filter. *Image* **2** (2014).

115. Bae, T.-W. Spatial and Temporal Bilateral Filter for Infrared Small Target Enhancement. *Infrared Phys. & Technol.* **63**, 42–53 (2014).

116. Knaus, C. & Zwicker, M. Dual-Domain Image Denoising. In *2013 IEEE International Conference on Image Processing*, 440–444 (IEEE, 2013).

117. Danielyan, A., Katkovnik, V. & Egiazarian, K. BM3D Frames and Variational Image Deblurring. *IEEE Transactions on Image Process.* **21**, 1715–1728 (2011).

118. Dabov, K., Foi, A., Katkovnik, V. & Egiazarian, K. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Process.* **16**, 2080–2095 (2007).

119. Jia, L. *et al.* Image Denoising via Sparse Representation Over Grouped Dictionaries with Adaptive Atom Size. *IEEE Access* **5**, 22514–22529 (2017).

120. Shao, L., Yan, R., Li, X. & Liu, Y. From Heuristic Optimization to Dictionary Learning: A Review and Comprehensive Comparison of Image Denoising Algorithms. *IEEE Transactions on Cybern.* **44**, 1001–1013 (2013).

121. Chatterjee, P. & Milanfar, P. Clustering-Based Denoising with Locally Learned Dictionaries. *IEEE Transactions on Image Process.* **18**, 1438–1451 (2009).

122. Aharon, M., Elad, M. & Bruckstein, A. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Process.* **54**, 4311–4322 (2006).

123. Elad, M. & Aharon, M. Image Denoising via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image processing* **15**, 3736–3745 (2006).

124. Pairis, S. *et al.* Shot-Noise-Limited Nanomechanical Detection and Radiation Pressure Backaction from an Electron Beam. *Phys. Rev. Lett.* **122**, 083603 (2019).

125. Seki, T., Ikuhara, Y. & Shibata, N. Theoretical Framework of Statistical Noise in Scanning Transmission Electron Microscopy. *Ultramicroscopy* **193**, 118–125 (2018).

126. Lee, Z., Rose, H., Lehtinen, O., Biskupek, J. & Kaiser, U. Electron Dose Dependence of Signal-to-Noise Ratio, Atom Contrast and Resolution in Transmission Electron Microscope Images. *Ultramicroscopy* **145**, 3–12 (2014).

127. Timischl, F., Date, M. & Nemoto, S. A Statistical Model of Signal–Noise in Scanning Electron Microscopy. *Scanning* **34**, 137–144 (2012).

128. Sim, K., Thong, J. & Phang, J. Effect of Shot Noise and Secondary Emission Noise in Scanning Electron Microscope Images. *Scanning: The J. Scanning Microsc.* **26**, 36–40 (2004).

129. Boyat, A. K. & Joshi, B. K. A Review Paper: Noise Models in Digital Image Processing. *arXiv preprint arXiv:1505.03489* (2015).

130. Meyer, R. R. & Kirkland, A. I. Characterisation of the Signal and Noise Transfer of CCD Cameras for Electron Detection. *Microsc. Res. Tech.* **49**, 269–280 (2000).

131. Kujawa, S. & Krahl, D. Performance of a Low-Noise CCD Camera Adapted to a Transmission Electron Microscope. *Ultramicroscopy* **46**, 395–403 (1992).

132. Rose, H. H. Optics of High-Performance Electron Microscopes. *Sci. Technol. Adv. Mater.* **9**, 014107 (2008).

133. Fujinaka, S., Sato, Y., Teranishi, R. & Kaneko, K. Understanding of Scanning-System Distortions of Atomic-Scale Scanning Transmission Electron Microscopy Images for Accurate Lattice Parameter Measurements. *J. Mater. Sci.* 1–11 (2020).

134. Sang, X. *et al.* Dynamic Scan Control in STEM: Spiral Scans. *Adv. Struct. Chem. Imaging* **2**, 1–8 (2016).

135. Ning, S. *et al.* Scanning Distortion Correction in STEM Images. *Ultramicroscopy* **184**, 274–283 (2018).

136. Ophus, C., Ciston, J. & Nelson, C. T. Correcting Nonlinear Drift Distortion of Scanning Probe and Scanning Transmission Electron Microscopies from Image Pairs with Orthogonal Scan Directions. *Ultramicroscopy* **162**, 1–9 (2016).

137. Jones, L. & Nellist, P. D. Identifying and Correcting Scan Noise and Drift in the Scanning Transmission Electron Microscope. *Microsc. Microanal.* **19**, 1050–1060 (2013).

138. Karthik, C., Kane, J., Butt, D. P., Windes, W. & Ubic, R. In Situ Transmission Electron Microscopy of Electron-Beam Induced Damage Process in Nuclear Grade Graphite. *J. nuclear materials* **412**, 321–326 (2011).

139. Roels, J. *et al.* An Interactive ImageJ Plugin for Semi-Automated Image Denoising in Electron Microscopy. *Nat. Commun.* **11**, 1–13 (2020).

140. Narasimha, R. *et al.* Evaluation of Denoising Algorithms for Biological Electron Tomography. *J. Struct. Biol.* **164**, 7–17 (2008).

141. Mevenkamp, N. *et al.* Poisson Noise Removal from High-Resolution STEM Images based on Periodic Block Matching. *Adv. Struct. Chem. Imaging* **1**, 3 (2015).

142. Bajić, B., Lindblad, J. & Sladoje, N. Blind Restoration of Images Degraded with Mixed Poisson-Gaussian Noise with Application in Transmission Electron Microscopy. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, 123–127 (IEEE, 2016).

143. Bodduna, K. & Weickert, J. Image Denoising with Less Artefacts: Novel Non-Linear Filtering on Fast Patch Reorderings. *arXiv preprint arXiv:2002.00638* (2020).

144. Jonić, S. *et al.* Denoising of High-Resolution Single-Particle Electron-Microscopy Density Maps by Their Approximation Using Three-Dimensional Gaussian Functions. *J. Struct. Biol.* **194**, 423–433 (2016).

145. Chung, S.-C. *et al.* Two-Stage Dimension Reduction for Noisy High-Dimensional Images and Application to Cryogenic Electron Microscopy. *arXiv* arXiv–1911 (2020).

146. Wang, J. & Yin, C. A Zernike-Moment-Based Non-Local Denoising Filter for Cryo-EM Images. *Sci. China Life Sci.* **56**, 384–390 (2013).

147. Furnival, T., Leary, R. K. & Midgley, P. A. Denoising Time-Resolved Microscopy Image Sequences with Singular Value Thresholding. *Ultramicroscopy* **178**, 112–124 (2017).

148. Sorzano, C. O. S., Ortiz, E., López, M. & Rodrigo, J. Improved Bayesian Image Denoising Based on Wavelets with Applications to Electron Microscopy. *Pattern Recognit.* **39**, 1205–1213 (2006).

149. Ouyang, J. *et al.* Cryo-Electron Microscope Image Denoising Based on the Geodesic Distance. *BMC Struct. Biol.* **18**, 18 (2018).

150. Du, H. A Nonlinear Filtering Algorithm for Denoising HR (S)TEM Micrographs. *Ultramicroscopy* **151**, 62–67 (2015).

151. Kushwaha, H. S., Tanwar, S., Rathore, K. & Srivastava, S. De-noising Filters for TEM (Transmission Electron Microscopy) Image of Nanomaterials. In *2012 Second International Conference on Advanced Computing & Communication Technologies*, 276–281 (IEEE, 2012).

152. Hanai, T., Morinaga, T., Suzuki, H. & Hibino, M. Maximum Entropy Restoration of Electron Microscope Images with a Random-Spatial-Distribution Constraint. *Scanning Microsc.* **11**, 379–390 (1997).

153. Pennycook, S. J. The Impact of STEM Aberration Correction on Materials Science. *Ultramicroscopy* **180**, 22–33 (2017).

154. Ramasse, Q. M. Twenty Years After: How "Aberration Correction in the STEM" Truly Placed a "A Synchrotron in a Microscope". *Ultramicroscopy* **180**, 41–51 (2017).

155. Hawkes, P. Aberration Correction Past and Present. *Philos. Transactions Royal Soc. A: Math. Phys. Eng. Sci.* **367**, 3637–3664 (2009).

156. Goodge, B. H., Bianco, E. & Kourkoutis, H. W. Atomic-Resolution Cryo-STEM Across Continuously Variable Temperature. *arXiv preprint arXiv:2001.11581* (2020).

157. Egerton, R. F. Radiation Damage to Organic and Inorganic Specimens in the TEM. *Micron* **119**, 72–87 (2019).

158. Egerton, R. F. Control of Radiation Damage in the TEM. *Ultramicroscopy* **127**, 100–108 (2013).

159. Egerton, R. Mechanisms of Radiation Damage in Beam-Sensitive Specimens, for TEM Accelerating Voltages Between 10 and 300 kV. *Microsc. Res. Tech.* **75**, 1550–1556 (2012).

160. Mankos, M. *et al.* Electron Optics for a Multi-Pass Transmission Electron Microscope. *Adv. Imaging Electron Phys.* **212** (2019).

161. Koppell, S. A. *et al.* Design for a 10 keV Multi-Pass Transmission Electron Microscope. *Ultramicroscopy* **207**, 112834 (2019).

162. Juffmann, T. *et al.* Multi-Pass Transmission Electron Microscopy. *Sci. Reports* **7**, 1–7 (2017).

163. Jones, L. *et al.* Managing Dose-, Damage- and Data-Rates in Multi-Frame Spectrum-Imaging. *Microscopy* **67**, i98–i113 (2018).

164. Krull, A., Buchholz, T.-O. & Jug, F. Noise2Void - Learning Denoising from Single Noisy Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2129–2137 (2019).

165. Guo, S., Yan, Z., Zhang, K., Zuo, W. & Zhang, L. Toward Convolutional Blind Denoising of Real Photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1712–1722 (2019).

166. Lefkimmiatis, S. Universal Denoising Networks: A Novel CNN Architecture for Image Denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3204–3213 (2018).

167. Weigert, M. *et al.* Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy. *Nat. Methods* **15**, 1090–1097 (2018).

168. Zhang, K., Zuo, W. & Zhang, L. FFDNet: Toward a Fast and Flexible Solution for CNN-Based Image Denoising. *IEEE Transactions on Image Process.* **27**, 4608–4622 (2018).

169. Weigert, M., Royer, L., Jug, F. & Myers, G. Isotropic Reconstruction of 3D Fluorescence Microscopy Images Using Convolutional Neural Networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 126–134 (Springer, 2017).

170. Zhang, K., Zuo, W., Chen, Y., Meng, D. & Zhang, L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Process.* **26**, 3142–3155 (2017).

171. Tai, Y., Yang, J., Liu, X. & Xu, C. MemNet: A Persistent Memory Network for Image Restoration. In *Proceedings of the IEEE International Conference on Computer Vision*, 4539–4547 (2017).

172. Mao, X., Shen, C. & Yang, Y.-B. Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections. In *Advances in Neural Information Processing Systems*, 2802–2810 (2016).

173. Buchholz, T.-O., Jordan, M., Pigino, G. & Jug, F. Cryo-CARE: Content-Aware Image Restoration for Cryo-Transmission Electron Microscopy Data. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 502–506 (IEEE, 2019).

174. Fang, L. *et al.* Deep Learning-Based Point-Scanning Super-Resolution Imaging. *bioRxiv* 740548 (2019).

175. Giannatou, E., Papavieros, G., Constantoudis, V., Papageorgiou, H. & Gogolides, E. Deep Learning Denoising of SEM Images Towards Noise-Reduced LER Measurements. *Microelectron. Eng.* **216**, 111051 (2019).

176. Chaudhary, N., Savari, S. A. & Yeddulapalli, S. S. Line Roughness Estimation and Poisson Denoising in Scanning Electron Microscope Images Using Deep Learning. *J. Micro/Nanolithography, MEMS, MOEMS* **18**, 024001 (2019).

177. Vasudevan, R. K. & Jesse, S. Deep Learning as a Tool for Image Denoising and Drift Correction. *Microsc. Microanal.* **25**, 190–191 (2019).

178. Wang, F., Henninen, T. R., Keller, D. & Erni, R. Noise2Atom: Unsupervised Denoising for Scanning Transmission Electron Microscopy Images. *Res. Sq.* DOI: 10.21203/rs.3.rs-54657/v1 (2020).

179. Bepler, T., Noble, A. J. & Berger, B. Topaz-Denoise: General Deep Denoising Models for CryoEM. *bioRxiv* 838920 (2019).

180. Lehtinen, J. *et al.* Noise2Noise: Learning Image Restoration without Clean Data. In *International Conference on Machine Learning*, 2965–2974 (2018).

181. Tegunov, D. & Cramer, P. Real-Time Cryo-Electron Microscopy Data Preprocessing with Warp. *Nat. Methods* **16**, 1146–1152 (2019).

182. Krishnan, A. P. *et al.* Optical aberration correction via phase diversity and deep learning. *bioRxiv* (2020).

183. Cumming, B. P. & Gu, M. Direct Determination of Aberration Functions in Microscopy by an Artificial Neural Network. *Opt. Express* **28**, 14511–14521 (2020).

184. Wang, W., Wu, B., Zhang, B., Li, X. & Tan, J. Correction of Refractive Index Mismatch-Induced Aberrations Under Radially Polarized Illumination by Deep Learning. *Opt. Express* **28**, 26028–26040 (2020).

185. Tian, Q. *et al.* DNN-Based Aberration Correction in a Wavefront Sensorless Adaptive Optics System. *Opt. Express* **27**, 10765–10776 (2019).

186. Rivenson, Y. *et al.* Deep Learning Enhanced Mobile-Phone Microscopy. *Acs Photonics* **5**, 2354–2364 (2018).

187. Nguyen, T. *et al.* Automatic Phase Aberration Compensation for Digital Holographic Microscopy Based on Deep Learning Background Detection. *Opt. Express* **25**, 15043–15057 (2017).

188. Jeon, S. & Kim, C. Deep Learning-Based Speed of Sound Aberration Correction in Photoacoustic Images. In *Photons Plus Ultrasound: Imaging and Sensing 2020*, vol. 11240, 112400J (International Society for Optics and Photonics, 2020).

189. Zhang, C., Berkels, B., Wirth, B. & Voyles, P. M. Joint Denoising and Distortion Correction for Atomic Column Detection in Scanning Transmission Electron Microscopy Images. *Microsc. Microanal.* **23**, 164–165 (2017).

190. Jin, P. & Li, X. Correction of Image Drift and Distortion in a Scanning Electron Microscopy. *J. Microsc.* **260**, 268–280 (2015).

191. Tong, X. *et al.* Image Registration with Fourier-Based Image Correlation: A Comprehensive Review of Developments and Applications. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **12**, 4062–4081 (2019).

192. Ede, J. M. & Beanland, R. Partial Scanning Transmission Electron Microscopy with Deep Learning. *Sci. Reports* **10**, 1–10 (2020).

193. Ede, J. M. Deep Learning Supersampled Scanning Transmission Electron Microscopy. *arXiv preprint arXiv:1910.10467* (2019).

194. Atta, R. E., Kasem, H. M. & Attia, M. A Comparison Study for Image Compression Based on Compressive Sensing. In *Eleventh International Conference on Graphics and Image Processing (ICGIP 2019)*, vol. 11373, 1137315 (International Society for Optics and Photonics, 2020).

195. Vidyasagar, M. *An Introduction to Compressed Sensing* (SIAM, 2019).

196. Rani, M., Dhok, S. B. & Deshmukh, R. A Systematic Review of Compressive Sensing: Concepts, Implementations and Applications. *IEEE Access* **6**, 4875–4894 (2018).

197. Eldar, Y. C. & Kutyniok, G. *Compressed Sensing: Theory and Applications* (Cambridge University Press, 2012).

198. Donoho, D. L. Compressed Sensing. *IEEE Transactions on Information Theory* **52**, 1289–1306 (2006).

199. Yuan, X. & Haimi-Cohen, R. Image Compression Based on Compressive Sensing: End-to-end Comparison with JPEG. *IEEE Transactions on Multimed.* (2020).

200. Gunasheela, S. & Prasantha, H. Compressed Sensing for Image Compression: Survey of Algorithms. In *Emerging Research in Computing, Information, Communication and Applications*, 507–517 (Springer, 2019).

201. Johnson, P. M., Recht, M. P. & Knoll, F. Improving the Speed of MRI with Artificial Intelligence. In *Seminars in Musculoskeletal Radiology*, vol. 24, 12 (NIH Public Access, 2020).

202. Ye, J. C. Compressed Sensing MRI: A Review from Signal Processing Perspective. *BMC Biomed. Eng.* **1**, 1–17 (2019).

203. Lustig, M., Donoho, D. & Pauly, J. M. Sparse MRI: The Application of Compressed Sensing for Rapid MR Imaging. *Magn. Reson. Medicine: An Off. J. Int. Soc. for Magn. Reson. Medicine* **58**, 1182–1195 (2007).

204. Shin, Y. J. *et al.* Low-Dose Abdominal CT Using a Deep Learning-Based Denoising Algorithm: A Comparison with CT Reconstructed with Filtered Back Projection or Iterative Reconstruction Algorithm. *Korean J. Radiol.* **21**, 356–364 (2020).

205. Cong, W. *et al.* Deep-Learning-Based Breast CT for Radiation Dose Reduction. In *Developments in X-Ray Tomography XII*, vol. 11113, 111131L (International Society for Optics and Photonics, 2019).

206. Barkan, O., Weill, J., Averbuch, A. & Dekel, S. Adaptive Compressed Tomography Sensing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2195–2202 (2013).

207. Almasri, F. & Debeir, O. Robust Perceptual Night Vision in Thermal Colorization. *arXiv preprint arXiv:2003.02204* (2020).

208. Chen, C., Chen, Q., Xu, J. & Koltun, V. Learning to See in the Dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3291–3300 (2018).

209. Peet, M. J., Henderson, R. & Russo, C. J. The Energy Dependence of Contrast and Damage in Electron Cryomicroscopy of Biological Molecules. *Ultramicroscopy* **203**, 125–131 (2019).

210. Zhang, X. *et al.* Radiation Damage in Nanostructured Materials. *Prog. Mater. Sci.* **96**, 217–321 (2018).

211. Lehnert, T., Lehtinen, O., Algara-Siller, G. & Kaiser, U. Electron Radiation Damage Mechanisms in 2D $MoSe_2$. *Appl. Phys. Lett.* **110**, 033106 (2017).

212. Hermannsdörfer, J., Tinnemann, V., Peckys, D. B. & de Jonge, N. The Effect of Electron Beam Irradiation in Environmental Scanning Transmission Electron Microscopy of Whole Cells in Liquid. *Microsc. Microanal.* **22**, 656–665 (2016).

213. Johnston-Peck, A. C., DuChene, J. S., Roberts, A. D., Wei, W. D. & Herzing, A. A. Dose-Rate-Dependent Damage of Cerium Dioxide in the Scanning Transmission Electron Microscope. *Ultramicroscopy* **170**, 1–9 (2016).

214. Jenkins, M. L. & Kirk, M. A. *Characterisation of Radiation Damage by Transmission Electron Microscopy* (CRC Press, 2000).

215. Egerton, R. F., Li, P. & Malac, M. Radiation Damage in the TEM and SEM. *Micron* **35**, 399–409 (2004).

216. S'ari, M., Cattle, J., Hondow, N., Brydson, R. & Brown, A. Low Dose Scanning Transmission Electron Microscopy of Organic Crystals by Scanning Moiré Fringes. *Micron* **120**, 1–9 (2019).

217. Mayoral, A., Mahugo, R., Sánchez-Sánchez, M. & Díaz, I. Cs-Corrected STEM Imaging of Both Pure and Silver-Supported Metal-Organic Framework MIL-100 (Fe). *ChemCatChem* **9**, 3497–3502 (2017).

218. Gnanasekaran, K., de With, G. & Friedrich, H. Quantification and Optimization of ADF-STEM Image Contrast for Beam-Sensitive Materials. *Royal Soc. Open Sci.* **5**, 171838 (2018).

219. Ilett, M., Brydson, R., Brown, A. & Hondow, N. Cryo-Analytical STEM of Frozen, Aqueous Dispersions of Nanoparticles. *Micron* **120**, 35–42 (2019).

220. Ede, J. M. Warwick Electron Microscopy Datasets. *Mach. Learn. Sci. Technol.* (2020).

221. Landau, H. J. Sampling, Data Transmission, and the Nyquist Rate. *Proc. IEEE* **55**, 1701–1706 (1967).

222. Amidror, I. Sub-Nyquist Artefacts and Sampling Moiré effects. *Royal Soc. Open Sci.* **2**, 140550 (2015).

223. Fadnavis, S. Image Interpolation Techniques in Digital Image Processing: An Overview. *Int. J. Eng. Res. Appl.* **4**, 70–73 (2014).

224. Getreuer, P. Linear Methods for Image Interpolation. *Image Process. On Line* **1**, 238–259 (2011).

225. Turkowski, K. Filters for Common Resampling Tasks. In *Graphics Gems*, 147–165 (Academic Press Professional, 1990).

226. Beretta, L. & Santaniello, A. Nearest Neighbor Imputation Algorithms: A Critical Evaluation. *BMC Med. Informatics Decis. Mak.* **16**, 74 (2016).

227. Alfeld, P. A Trivariate Clough—Tocher Scheme for Tetrahedral Data. *Comput. Aided Geom. Des.* **1**, 169–181 (1984).

228. Cruz, C., Mehta, R., Katkovnik, V. & Egiazarian, K. O. Single Image Super-Resolution Based on Wiener Filter in Similarity Domain. *IEEE Transactions on Image Process.* **27**, 1376–1389 (2017).

229. Zulkifli, N., Karim, S., Shafie, A. & Sarfraz, M. Rational Bicubic Ball for Image Interpolation. In *Journal of Physics: Conference Series*, vol. 1366, 012097 (IOP Publishing, 2019).

230. Costella, J. The Magic Kernel. Towards Data Science, Online: https://web.archive.org/web/20170707165835/http://johncostella.webs.com/magic (2017).

231. Olivier, R. & Hanqiang, C. Nearest Neighbor Value Interpolation. *Int. J. Adv. Comput. Sci. Appl.* **3**, 25–30 (2012).

232. Jones, L. *et al.* Managing Dose-, Damage- and Data-Rates in Multi-Frame Spectrum-Imaging. *Microscopy* **67**, i98–i113 (2018).

233. Trampert, P. *et al.* How Should a Fixed Budget of Dwell Time be Spent in Scanning Electron Microscopy to Optimize Image Quality? *Ultramicroscopy* **191**, 11–17 (2018).

234. Stevens, A. *et al.* A Sub-Sampled Approach to Extremely Low-Dose STEM. *Appl. Phys. Lett.* **112**, 043104 (2018).

235. Hwang, S., Han, C. W., Venkatakrishnan, S. V., Bouman, C. A. & Ortalan, V. Towards the Low-Dose Characterization of Beam Sensitive Nanostructures via Implementation of Sparse Image Acquisition in Scanning Transmission Electron Microscopy. *Meas. Sci. Technol.* **28**, 045402 (2017).

236. Hujsak, K., Myers, B. D., Roth, E., Li, Y. & Dravid, V. P. Suppressing Electron Exposure Artifacts: An Electron Scanning Paradigm with Bayesian Machine Learning. *Microsc. Microanal.* **22**, 778–788 (2016).

237. Anderson, H. S., Ilic-Helms, J., Rohrer, B., Wheeler, J. & Larson, K. Sparse Imaging for Fast Electron Microscopy. In *Computational Imaging XI*, vol. 8657, 86570C (International Society for Optics and Photonics, 2013).

238. Stevens, A., Yang, H., Carin, L., Arslan, I. & Browning, N. D. The Potential for Bayesian Compressive Sensing to Significantly Reduce Electron Dose in High-Resolution STEM Images. *Microscopy* **63**, 41–51 (2013).

239. Candes, E. & Romberg, J. Sparsity and Incoherence in Compressive Sampling. *Inverse Probl.* **23**, 969 (2007).

240. Kovarik, L., Stevens, A., Liyu, A. & Browning, N. D. Implementing an Accurate and Rapid Sparse Sampling Approach for Low-Dose Atomic Resolution STEM Imaging. *Appl. Phys. Lett.* **109**, 164102 (2016).

241. Sang, X. *et al.* Dynamic Scan Control in STEM: Spiral Scans. *Adv. Struct. Chem. Imaging* **2**, 6 (2017).

242. Béché, A., Goris, B., Freitag, B. & Verbeeck, J. Development of a Fast Electromagnetic Beam Blanker for Compressed Sensing in Scanning Transmission Electron Microscopy. *Appl. Phys. Lett.* **108**, 093103 (2016).

243. Li, X., Dyck, O., Kalinin, S. V. & Jesse, S. Compressed Sensing of Scanning Transmission Electron Microscopy (STEM) with Nonrectangular Scans. *Microsc. Microanal.* **24**, 623–633 (2018).

244. Sang, X. *et al.* Precision Controlled Atomic Resolution Scanning Transmission Electron Microscopy using Spiral Scan Pathways. *Sci. Reports* **7**, 43585 (2017).

245. Gandhare, S. & Karthikeyan, B. Survey on FPGA Architecture and Recent Applications. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 1–4 (IEEE, 2019).

246. Qiao, M., Meng, Z., Ma, J. & Yuan, X. Deep Learning for Video Compressive Sensing. *APL Photonics* **5**, 030801 (2020).

247. Wu, Y., Rosca, M. & Lillicrap, T. Deep Compressed Sensing. *arXiv preprint arXiv:1905.06723* (2019).

248. Adler, A., Boublil, D. & Zibulevsky, M. Block-Based Compressed Sensing of Images via Deep Learning. In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, 1–6 (IEEE, 2017).

249. de Haan, K., Ballard, Z. S., Rivenson, Y., Wu, Y. & Ozcan, A. Resolution Enhancement in Scanning Electron Microscopy Using Deep Learning. *Sci. Reports* **9**, 1–7 (2019).

250. Gao, Z., Ma, W., Huang, S., Hua, P. & Lan, C. Deep Learning for Super-Resolution in a Field Emission Scanning Electron Microscope. *AI* **1**, 1–10 (2020).

251. Ede, J. M. & Beanland, R. Adaptive Learning Rate Clipping Stabilizes Learning. *Mach. Learn. Sci. Technol.* **1**, 015011 (2020).

252. Suveer, A., Gupta, A., Kylberg, G. & Sintorn, I.-M. Super-Resolution Reconstruction of Transmission Electron Microscopy Images Using Deep Learning. In *2019 IEEE 16th International Symposium on Biomedical Imaging*, 548–551 (IEEE, 2019).

253. Ahmed, M. W. & Abdulla, A. A. Quality Improvement for Exemplar-based Image Inpainting Using a Modified Searching Mechanism. *UHD J. Sci. Technol.* **4**, 1–8 (2020).

254. Pinjarkar, A. V. & Tuptewar, D. Robust Exemplar-Based Image and Video Inpainting for Object Removal and Region Filling. In *Computing, Communication and Signal Processing*, 817–825 (Springer, 2019).

255. Zhang, N., Ji, H., Liu, L. & Wang, G. Exemplar-Based Image Inpainting Using Angle-Aware Patch Matching. *EURASIP J. on Image Video Process.* **2019**, 70 (2019).

256. Criminisi, A., Pérez, P. & Toyama, K. Region Filling and Object Removal by Exemplar-Based Image Inpainting. *IEEE Transactions on Image Process.* **13**, 1200–1212 (2004).

257. Lu, M. & Niu, S. A Detection Approach Using LSTM-CNN for Object Removal Caused by Exemplar-Based Image Inpainting. *Electronics* **9**, 858 (2020).

258. Telea, A. An Image Inpainting Technique Based on the Fast Marching Method. *J. Graph. Tools* **9**, 23–34 (2004).

259. Bertalmio, M., Bertozzi, A. L. & Sapiro, G. Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, I–I (IEEE, 2001).

260. He, T. *et al.* Bag of Tricks for Image Classification with Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 558–567 (2019).

261. Sun, Y., Xue, B., Zhang, M. & Yen, G. G. Evolving Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evol. Comput.* **24**, 394–407 (2019).

262. Rawat, W. & Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* **29**, 2352–2449 (2017).

263. Druzhkov, P. N. & Kustikova, V. D. A Survey of Deep Learning Methods and Software Tools for Image Classification and Object Detection. *Pattern Recognit. Image Analysis* **26**, 9–15 (2016).

264. Yokoyama, Y. *et al.* Development of a Deep Learning-Based Method to Identify "Good" Regions of a Cryo-Electron Microscopy Grid. *Biophys. Rev.* 1–6 (2020).

265. Aguiar, J., Gong, M., Unocic, R., Tasdizen, T. & Miller, B. Decoding Crystallography from High-Resolution Electron Imaging and Diffraction Datasets with Deep Learning. *Sci. Adv.* **5**, eaaw1949 (2019).

266. Vasudevan, R. K. *et al.* Mapping Mesoscopic Phase Evolution During E-Beam Induced Transformations via Deep Learning of Atomically Resolved Images. *npj Comput. Mater.* **4**, 1–9 (2018).

267. Avramov, T. K. *et al.* Deep Learning for Validating and Estimating Resolution of Cryo-Electron Microscopy Density Maps. *Molecules* **24**, 1181 (2019).

268. Koch, G., Zemel, R. & Salakhutdinov, R. Siamese Neural Networks for One-Shot Image Recognition. In *ICML Deep Learning Workshop*, vol. 2 (Lille, 2015).

269. Chopra, S., Hadsell, R. & LeCun, Y. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 539–546 (IEEE, 2005).

270. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. & Shah, R. Signature Verification Using a "Siamese" Time Delay Neural Network. In *Advances in Neural Information Processing Systems*, 737–744 (1994).

271. Cai, Q., Pan, Y., Yao, T., Yan, C. & Mei, T. Memory Matching Networks for One-Shot Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4080–4088 (2018).

272. Li, X. *et al.* Predicting the Effective Mechanical Property of Heterogeneous Materials by Image Based Modeling and Deep Learning. *Comput. Methods Appl. Mech. Eng.* **347**, 735–753 (2019).

273. Sanchez-Garcia, R., Segura, J., Maluenda, D., Carazo, J. M. & Sorzano, C. O. S. Deep Consensus, A Deep Learning-Based Approach for Particle Pruning in Cryo-Electron Microscopy. *IUCrJ* **5**, 854–865 (2018).

274. Wang, F. *et al.* DeepPicker: A Deep Learning Approach for Fully Automated Particle Picking in Cryo-EM. *J. Struct. Biol.* **195**, 325–336 (2016).

275. George, B. *et al.* CASSPER: A Semantic Segmentation Based Particle Picking Algorithm for Single Particle Cryo-Electron Microscopy. *bioRxiv* (2020).

276. Roberts, G. *et al.* Deep Learning for Semantic Segmentation of Defects in Advanced STEM Images of Steels. *Sci. Reports* **9**, 1–12 (2019).

277. Madsen, J. *et al.* A Deep Learning Approach to Identify Local Structures in Atomic-Resolution Transmission Electron Microscopy Images. *Adv. Theory Simulations* **1**, 1800037 (2018).

278. Ziatdinov, M. *et al.* Deep Learning of Atomically Resolved Scanning Transmission Electron Microscopy Images: Chemical Identification and Tracking Local Transformations. *ACS Nano* **11**, 12742–12752 (2017).

279. Ziatdinov, M. *et al.* Building and Exploring Libraries of Atomic Defects in Graphene: Scanning Transmission Electron and Scanning Tunneling Microscopy Study. *Sci. Adv.* **5**, eaaw8989 (2019).

280. Meyer, J. C. *et al.* Direct Imaging of Lattice Atoms and Topological Defects in Graphene Membranes. *Nano Lett.* **8**, 3582–3586 (2008).

281. Meyer, J. C. *et al.* Experimental Analysis of Charge Redistribution Due to Chemical Bonding by High-Resolution Transmission Electron Microscopy. *Nat. Mater.* **10**, 209–215 (2011).

282. He, X. *et al.* In Situ Atom Scale Visualization of Domain Wall Dynamics in VO_2 Insulator-Metal Phase Transition. *Sci. Reports* **4**, 6544 (2014).

283. Nagao, K., Inuzuka, T., Nishimoto, K. & Edagawa, K. Experimental Observation of Quasicrystal Growth. *Phys. Rev. Lett.* **115**, 075501 (2015).

284. Li, X. *et al.* Direct Observation of the Layer-by-Layer Growth of ZnO Nanopillar by In Situ High Resolution Transmission Electron Microscopy. *Sci. Reports* **7**, 40911 (2017).

285. Schneider, S., Surrey, A., Pohl, D., Schultz, L. & Rellinghaus, B. Atomic Surface Diffusion on Pt Nanoparticles Quantified by High-Resolution Transmission Electron Microscopy. *Micron* **63**, 52–56 (2014).

286. Hussaini, Z., Lin, P. A., Natarajan, B., Zhu, W. & Sharma, R. Determination of Atomic Positions from Time Resolved High Resolution Transmission Electron Microscopy Images. *Ultramicroscopy* **186**, 139–145 (2018).

287. Pham, D. L., Xu, C. & Prince, J. L. Current Methods in Medical Image Segmentation. *Annu. Rev. Biomed. Eng.* **2**, 315–337 (2000).

288. Mesejo, P., Valsecchi, A., Marrakchi-Kacem, L., Cagnoni, S. & Damas, S. Biomedical Image Segmentation Using Geometric Deformable Models and Metaheuristics. *Comput. Med. Imaging Graph.* **43**, 167–178 (2015).

289. Zheng, Y., Jeon, B., Xu, D., Wu, Q. M. & Zhang, H. Image Segmentation by Generalized Hierarchical Fuzzy C-Means Algorithm. *J. Intell. & Fuzzy Syst.* **28**, 961–973 (2015).

290. Hao, S., Zhou, Y. & Guo, Y. A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing* (2020).

291. Sultana, F., Sufian, A. & Dutta, P. Evolution of Image Segmentation Using Deep Convolutional Neural Network: A Survey. *Knowledge-Based Syst.* 106062 (2020).

292. Minaee, S. *et al.* Image segmentation Using deep learning: A survey. *arXiv preprint arXiv:2001.05566* (2020).

293. Guo, Y., Liu, Y., Georgiou, T. & Lew, M. S. A Review of Semantic Segmentation using Deep Neural Networks. *Int. J. Multimed. Inf. Retr.* **7**, 87–93 (2018).

294. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. & Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 801–818 (2018).

295. Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587* (2017).

296. Badrinarayanan, V., Kendall, A. & Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis Mach. Intell.* **39**, 2481–2495 (2017).

297. Ronneberger, O., Fischer, P. & Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 234–241 (Springer, 2015).

298. Yi, J., Yuan, Z. & Peng, J. Adversarial-Prediction Guided Multi-Task Adaptation for Semantic Segmentation of Electron Microscopy Images. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 1205–1208 (IEEE, 2020).

299. Khadangi, A., Boudier, T. & Rajagopal, V. EM-net: Deep Learning for Electron Microscopy Image Segmentation. *bioRxiv* (2020).

300. Roels, J. & Saeys, Y. Cost-Efficient Segmentation of Electron Microscopy Images Using Active Learning. *arXiv preprint arXiv:1911.05548* (2019).

301. Yu, Z. X. *et al.* High-Throughput, Algorithmic Determination of Pore Parameters from Electron Microscopy. *Comput. Mater. Sci.* **171**, 109216 (2020).

302. Fakhry, A., Zeng, T. & Ji, S. Residual Deconvolutional Networks for Brain Electron Microscopy Image Segmentation. *IEEE Transactions on Med. Imaging* **36**, 447–456 (2016).

303. Urakubo, H., Bullmann, T., Kubota, Y., Oba, S. & Ishii, S. UNI-EM: An Environment for Deep Neural Network-Based Automated Segmentation of Neuronal Electron Microscopic Images. *Sci. Reports* **9**, 1–9 (2019).

304. Roberts, G. *et al.* DefectNet – A Deep Convolutional Neural Network for Semantic Segmentation of Crystallographic Defects in Advanced Microscopy Images. *Microsc. Microanal.* **25**, 164–165 (2019).

305. Ibtehaz, N. & Rahman, M. S. MultiResUNet: Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation. *Neural Networks* **121**, 74–87 (2020).

306. Groschner, C. K., Choi, C. & Scott, M. Methodologies for Successful Segmentation of HRTEM Images via Neural Network. *arXiv preprint arXiv:2001.05022* (2020).

307. Horwath, J. P., Zakharov, D. N., Megret, R. & Stach, E. A. Understanding Important Features of Deep Learning Models for Transmission Electron Microscopy Image Segmentation. *arXiv preprint arXiv:1912.06077* (2019).

308. Feng, D. *et al.* Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intell. Transp. Syst.* (2020).

309. Yang, K., Bi, S. & Dong, M. Lightningnet: Fast and Accurate Semantic Segmentation for Autonomous Driving Based on 3D LIDAR Point Cloud. In *2020 IEEE International Conference on Multimedia and Expo*, 1–6 (IEEE, 2020).

310. Hofmarcher, M. *et al.* Visual Scene Understanding for Autonomous Driving Using Semantic Segmentation. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 285–296 (Springer, 2019).

311. Blum, H., Sarlin, P.-E., Nieto, J., Siegwart, R. & Cadena, C. Fishyscapes: A Benchmark for Safe Semantic Segmentation in Autonomous Driving. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2019).

312. Zhou, W., Berrio, J. S., Worrall, S. & Nebot, E. Automated Evaluation of Semantic Segmentation Robustness for Autonomous Driving. *IEEE Transactions on Intell. Transp. Syst.* **21**, 1951–1963 (2019).

313. Pfisterer, K. J. *et al.* Fully-Automatic Semantic Segmentation for Food Intake Tracking in Long-Term Care Homes. *arXiv preprint arXiv:1910.11250* (2019).

314. Aslan, S., Ciocca, G. & Schettini, R. Semantic Food Segmentation for Automatic Dietary Monitoring. In *2018 IEEE 8th International Conference on Consumer Electronics-Berlin*, 1–6 (IEEE, 2018).

315. Ghosh, S., Ray, N., Boulanger, P., Punithakumar, K. & Noga, M. Automated Left Atrial Segmentation from Magnetic Resonance Image Sequences Using Deep Convolutional Neural Network with Autoencoder. In *2020 IEEE 17th International Symposium on Biomedical Imaging*, 1756–1760 (IEEE, 2020).

316. Memis, A., Varli, S. & Bilgili, F. Semantic Segmentation of the Multiform Proximal Femur and Femoral Head Bones with the Deep Convolutional Neural Networks in Low Quality MRI Sections Acquired in Different MRI Protocols. *Comput. Med. Imaging Graph.* 101715 (2020).

317. Duran, A., Jodoin, P.-M. & Lartizien, C. Prostate Cancer Semantic Segmentation by Gleason Score Group in mp-MRI with Self Attention Model on the Peripheral Zone. In *Medical Imaging with Deep Learning* (2020).

318. Bevilacqua, V. *et al.* A Comparison Between Two Semantic Deep Learning Frameworks for the Autosomal Dominant Polycystic Kidney Disease Segmentation Based on Magnetic Resonance Images. *BMC Med. Informatics Decis. Mak.* **19**, 1–12 (2019).

319. Liu, F. *et al.* Deep Convolutional Neural Network and 3D Deformable Approach for Tissue Segmentation in Musculoskeletal Magnetic Resonance Imaging. *Magn. Reson. Medicine* **79**, 2379–2391 (2018).

320. Taghanaki, S. A., Abhishek, K., Cohen, J. P., Cohen-Adad, J. & Hamarneh, G. Deep Semantic Segmentation of Natural and Medical Images: A Review. *Artif. Intell. Rev.* 1–42 (2020).

321. Tajbakhsh, N. *et al.* Embracing Imperfect Datasets: A Review of Deep Learning Solutions for Medical Image Segmentation. *Med. Image Analysis* 101693 (2020).

322. Du, G., Cao, X., Liang, J., Chen, X. & Zhan, Y. Medical Image Segmentation Based on U-Net: A Review. *J. Imaging Sci. Technol.* **64**, 20508–1 (2020).

323. Yang, X. *et al.* Hybrid Attention for Automatic Segmentation of Whole Fetal Head in Prenatal Ultrasound Volumes. *Comput. Methods Programs Biomed.* 105519 (2020).

324. Wang, X. *et al.* Joint Segmentation and Landmark Localization of Fetal Femur in Ultrasound Volumes. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, 1–5 (IEEE, 2019).

325. Venturini, L., Papageorghiou, A. T., Noble, J. A. & Namburete, A. I. Multi-task CNN for Structural Semantic Segmentation in 3D Fetal Brain Ultrasound. In *Annual Conference on Medical Image Understanding and Analysis*, 164–173 (Springer, 2019).

326. Yang, X. *et al.* Towards Automated Semantic Segmentation in Prenatal Volumetric Ultrasound. *IEEE Transactions on Med. Imaging* **38**, 180–193 (2018).

327. Tasar, O., Tarabalka, Y., Giros, A., Alliez, P. & Clerc, S. StandardGAN: Multi-source Domain Adaptation for Semantic Segmentation of Very High Resolution Satellite Images by Data Standardization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 192–193 (2020).

328. Barthakur, M. & Sarma, K. K. Deep Learning Based Semantic Segmentation Applied to Satellite Image. In *Data Visualization and Knowledge Engineering*, 79–107 (Springer, 2020).

329. Wu, M., Zhang, C., Liu, J., Zhou, L. & Li, X. Towards Accurate High Resolution Satellite Image Semantic Segmentation. *IEEE Access* **7**, 55609–55619 (2019).

330. Wurm, M., Stark, T., Zhu, X. X., Weigand, M. & Taubenböck, H. Semantic Segmentation of Slums in Satellite Images Using Transfer Learning on Fully Convolutional Neural Networks. *ISPRS J. Photogramm. Remote. Sens.* **150**, 59–69 (2019).

331. Zhou, L., Zhang, C. & Wu, M. D-LinkNet: LinkNet With Pretrained Encoder and Dilated Convolution for High Resolution Satellite Imagery Road Extraction. In *CVPR Workshops*, 182–186 (2018).

332. Joyce, T., Chartsias, A. & Tsaftaris, S. A. Deep Multi-Class Segmentation Without Ground-Truth Labels. In *1st Conference on Medical Imaging with Deep Learning* (2018).

333. Araslanov, N. & Roth, S. Single-Stage Semantic Segmentation from Image Labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4253–4262 (2020).

334. Chen, Z., Tian, Z., Li, X., Zhang, Y. & Dormer, J. D. Exploiting Confident Information for Weakly Supervised Prostate Segmentation Based on Image-Level Labels. In *Medical Imaging 2020: Image-Guided Procedures, Robotic Interventions, and Modeling*, vol. 11315, 1131523 (International Society for Optics and Photonics, 2020).

335. Jing, L., Chen, Y. & Tian, Y. Coarse-to-Fine Semantic Segmentation from Image-Level Labels. *IEEE Transactions on Image Process.* **29**, 225–236 (2019).

336. Oh, S. J. *et al.* Exploiting Saliency for Object Segmentation from Image Level Labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 5038–5047 (IEEE, 2017).

337. Ede, J. M., Peters, J. J. P., Sloan, J. & Beanland, R. Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. *arXiv preprint arXiv:2001.10938* (2020).

338. Frabboni, S., Gazzadi, G. C. & Pozzi, G. Young's Double-Slit Interference Experiment with Electrons. *Am. J. Phys.* **75**, 1053–1055 (2007).

339. Matteucci, G. & Beeli, C. An Experiment on Electron Wave-Particle Duality Including a Planck Constant Measurement. *Am. J. Phys.* **66**, 1055–1059 (1998).

340. Lehmann, M. & Lichte, H. Tutorial on Off-Axis Electron Holography. *Microsc. Microanal.* **8**, 447–466 (2002).

341. Tonomura, A. Applications of Electron Holography. *Rev. Mod. Phys.* **59**, 639 (1987).

342. Lentzen, M. & Urban, K. Reconstruction of the Projected Crystal Potential in Transmission Electron Microscopy by Means of a Maximum-Likelihood Refinement Algorithm. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **56**, 235–247 (2000).

343. Auslender, A., Halabi, M., Levi, G., Diéguez, O. & Kohn, A. Measuring the Mean Inner Potential of Al_2O_3 Sapphire Using Off-Axis Electron Holography. *Ultramicroscopy* **198**, 18–25 (2019).

344. Fu, Q., Lichte, H. & Völkl, E. Correction of Aberrations of an Electron Microscope by Means of Electron Holography. *Phys. Rev. Lett.* **67**, 2319 (1991).

345. McCartney, M. R. & Gajdardziska-Josifovska, M. Absolute Measurement of Normalized Thickness, t/λ_i, from Off-Axis Electron Holography. *Ultramicroscopy* **53**, 283–289 (1994).

346. Park, H. S. *et al.* Observation of the Magnetic Flux and Three-Dimensional Structure of Skyrmion Lattices by Electron Holography. *Nat. Nanotechnol.* **9**, 337–342 (2014).

347. Dunin-Borkowski, R. E. *et al.* Off-Axis Electron Holography of Magnetic Nanowires and Chains, Rings, and Planar Arrays of Magnetic Nanoparticles. *Microsc. Res. Tech.* **64**, 390–402 (2004).

348. Lubk, A. *et al.* Fundamentals of Focal Series Inline Electron Holography. In *Advances in Imaging and Electron Physics*, vol. 197, 105–147 (Elsevier, 2016).

349. Koch, C. T. Towards Full-Resolution Inline Electron Holography. *Micron* **63**, 69–75 (2014).

350. Haigh, S. J., Jiang, B., Alloyeau, D., Kisielowski, C. & Kirkland, A. I. Recording Low and High Spatial Frequencies in Exit Wave Reconstructions. *Ultramicroscopy* **133**, 26–34 (2013).

351. Koch, C. T. & Lubk, A. Off-Axis and Inline Electron Holography: A Quantitative Comparison. *Ultramicroscopy* **110**, 460–471 (2010).

352. Van Dyck, D., de Beeck, M. O. & Coene, W. Object Wavefunction Reconstruction in High Resolution Electron Microscopy. In *Proceedings of 1st International Conference on Image Processing*, vol. 3, 295–298 (IEEE, 1994).

353. Ozsoy-Keskinbora, C., Boothroyd, C., Dunin-Borkowski, R., Van Aken, P. & Koch, C. Hybridization Approach to In-Line and Off-Axis (Electron) Holography for Superior Resolution and Phase Sensitivity. *Sci. Reports* **4**, 1–10 (2014).

354. Rivenson, Y., Zhang, Y., Günaydın, H., Teng, D. & Ozcan, A. Phase Recovery and Holographic Image Reconstruction Using Deep Learning in Neural Networks. *Light. Sci. & Appl.* **7**, 17141–17141 (2018).

355. Wu, Y. *et al.* Extended Depth-of-Field in Holographic Imaging Using Deep-Learning-Based AutofocUsing and Phase Recovery. *Optica* **5**, 704–710 (2018).

356. Sinha, A., Lee, J., Li, S. & Barbastathis, G. Lensless Computational Imaging Through Deep Learning. *Optica* **4**, 1117–1125 (2017).

357. Beach, M. J. *et al.* QuCumber: Wavefunction Reconstruction with Neural Networks. *arXiv preprint arXiv:1812.09329* (2018).

358. Dral, P. O. Quantum Chemistry in the Age of Machine Learning. *The J. Phys. Chem. Lett.* **11**, 2336–2347 (2020).

359. Liu, X. *et al.* Deep Learning for Feynman's Path Integral in Strong-Field Time-Dependent Dynamics. *Phys. Rev. Lett.* **124**, 113202 (2020).

360. Bharti, K., Haug, T., Vedral, V. & Kwek, L.-C. Machine Learning Meets Quantum Foundations: A Brief Survey. *arXiv preprint arXiv:2003.11224* (2020).

361. Carleo, G. *et al.* NetKet: A Machine Learning Toolkit for Many-Body Quantum Systems. *arXiv preprint arXiv:1904.00031* (2019).

362. Schütt, K., Gastegger, M., Tkatchenko, A., Müller, K.-R. & Maurer, R. J. Unifying Machine Learning and Quantum Chemistry with a Deep Neural Network for Molecular Wavefunctions. *Nat. Commun.* **10**, 1–10 (2019).

363. Laanait, N., He, Q. & Borisevich, A. Y. Reconstruction of 3-D Atomic Distortions from Electron Microscopy with Deep Learning. *arXiv preprint arXiv:1902.06876* (2019).

364. Morgan, A. J., Martin, A. V., D'Alfonso, A. J., Putkunz, C. T. & Allen, L. J. Direct Exit-Wave Reconstruction From a Single Defocused Image. *Ultramicroscopy* **111**, 1455–1460 (2011).

365. Martin, A. & Allen, L. Direct Retrieval of a Complex Wave From its Diffraction Pattern. *Opt. communications* **281**, 5114–5121 (2008).

366. Schlitz, M. Science Without Publication Paywalls a Preamble to: cOAlition S for the Realisation of Full and Immediate Open Access. *Sci. Eur.* (2018).

367. Coalition of European Funders Announces "Plan S" to Require Full OA, Cap APCs, & Disallow Publication in Hybrid Journals. SPARC, Online: https://sparcopen.org/news/2018/coalition-european-funders-announces-plan-s (2018).

368. cOAlition S. Plan S: Making Full and Immediate Open Access a Reality. Online: https://www.coalition-s.org (2020).

369. Banks, G. C. *et al.* Answers to 18 Questions About Open Science Practices. *J. Bus. Psychol.* **34**, 257–270 (2019).

370. Shi, R. *et al.* FTDL: An FPGA-Tailored Architecture for Deep Learning Systems. In *FPGA*, 320 (2020).

371. Kaarmukilan, S., Poddar, S. *et al.* FPGA Based Deep Learning Models for Object Detection and Recognition Comparison of Object Detection Comparison of Object Detection Models Using FPGA. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, 471–474 (IEEE, 2020).

372. Wang, T., Wang, C., Zhou, X. & Chen, H. An Overview of FPGA Based Deep Learning Accelerators: Challenges and Opportunities. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 1674–1681 (IEEE, 2019).

373. Guo, K., Zeng, S., Yu, J., Wang, Y. & Yang, H. [DL] A Survey of FPGA-Based Neural Network Inference Accelerators. *ACM Transactions on Reconfigurable Technol. Syst. (TRETS)* **12**, 1–26 (2019).

374. Cano, A. A Survey on Graphic Processing Unit Computing for Large-Scale Data Mining. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **8**, e1232 (2018).

375. Nvidia. Tesla V100 GPU Architecture Whitepaper. Online: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf (2017).

376. Gaster, B. R. *Heterogeneous Computing with OpenCL, 2nd Edition* (Elsevier/Morgan Kaufmann, 2013).

377. Gordienko, Y. *et al.* Scaling Analysis of Specialized Tensor Processing Architectures for Deep Learning Models. In *Deep Learning: Concepts and Architectures*, 65–99 (Springer, 2020).

378. Jouppi, N., Young, C., Patil, N. & Patterson, D. Motivation for and Evaluation of the First Tensor Processing Unit. *IEEE Micro* **38**, 10–19 (2018).

379. Jouppi, N. P. *et al.* In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1–12 (2017).

380. Mattson, P. *et al.* MLPerf Training Benchmark. *arXiv preprint arXiv:1910.01500* (2020).

381. MLPerf: Fair and Useful Benchmarks for Measuring Training and Inference Performance of ML Hardware, Software, and Services. Online: https://mlperf.org (2020).

382. Wang, Y. E., Wei, G.-Y. & Brooks, D. Benchmarking TPU, GPU, and CPU Platforms for Deep Learning. *arXiv preprint arXiv:1907.10701* (2019).

383. Wang, Y. *et al.* Performance and Power Evaluation of AI Accelerators for Training Deep Learning Models. *arXiv preprint arXiv:1909.06842* (2019).

384. Li, F., Ye, Y., Tian, Z. & Zhang, X. Cpu versus gpu: Which can perform matrix computation faster – performance comparison for basic linear algebra subprograms. *Neural Comput. Appl.* **31**, 4353–4365 (2019).

385. Awan, A. A., Subramoni, H. & Panda, D. K. An In-Depth Performance Characterization of CPU-and GPU-Based DNN Training on Modern Architectures. In *Proceedings of the Machine Learning on HPC Environments*, 1–8 (2017).

386. Nurvitadhi, E. *et al.* Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 5–14 (2017).

387. GPU vs FPGA Performance Comparison. Berten Digital Signal Processing, Online: http://www.bertendsp.com/pdf/whitepaper/BWP001_GPU_vs_FPGA_Performance_Comparison_v1.0.pdf (2016).

388. Nangia, R. & Shukla, N. K. Resource Utilization Optimization with Design Alternatives in FPGA Based Arithmetic Logic Unit Architectures. *Procedia Comput. Sci.* **132**, 843–848 (2018).

389. Grover, N. & Soni, M. Design of fpga based 32-bit floating point arithmetic unit and verification of its vhdl code using matlab. *Int. J. Inf. Eng. Electron. Bus.* **6**, 1 (2014).

390. Dolbeau, R. Theoretical Peak FLOPS Per Instruction Set: A Tutorial. *The J. Supercomput.* **74**, 1341–1377 (2018).

391. Strubell, E., Ganesh, A. & McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).

392. Nelson, M. J. & Hoover, A. K. Notes on Using Google Colaboratory in AI Education. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 533–534 (2020).

393. Bisong, E. Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, 59–64 (Springer, 2019).

394. Tutorialspoint. Colab Tutorial. Online: https://www.tutorialspoint.com/google_colab/google_colab_tutorial.pdf (2019).

395. Carneiro, T. *et al.* Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access* **6**, 61677–61685 (2018).

396. Kaggle Documentation. Online: https://www.kaggle.com/docs (2020).

397. Kalinin, S. V., Vasudevan, R. K. & Ziatdinov, M. Decoding the Relationship Between Domain Structure and Functionality in Ferroelectrics via Hidden Latent Variables. *arXiv preprint arXiv:2006.01374* (2020).

398. Green, O. How to Install a New Graphics Card – From Hardware to Drivers. Help Desk Geek, Online: https://helpdeskgeek.com/how-to/how-to-install-a-new-graphics-card-from-hardware-to-drivers (2019).

399. Ryan, T. How to Install a Graphics Card. PC World, Online: https://www.pcworld.com/article/2913370/how-to-install-a-graphics-card.html (2017).

400. Radecic, D. An Utterly Simple Guide on Installing Tensorflow-GPU 2.0 on Windows 10. Towards Data Science, Online: https://towardsdatascience.com/an-utterly-simple-guide-on-installing-tensorflow-gpu-2-0-on-windows-10-198368dc07a1 (2020).

401. Varile, M. Train Neural Networks Using AMD GPU and Keras. Towards Data Science, Online: https://towardsdatascience.com/train-neural-networks-Using-amd-gpus-and-keras-37189c453878 (2019).

402. Tim Dettmers. A Full Hardware Guide to Deep Learning. Online: https://timdettmers.com/2018/12/16/deep-learning-hardware-guide (2018).

403. Chetlur, S. *et al.* cuDNN: Efficient Primitives for Deep Learning. *arXiv preprint arXiv:1410.0759* (2014).

404. List of Cloud Services for Deep Learning. Online: https://github.com/zszazi/Deep-learning-in-cloud (2020).

405. Marozzo, F. Infrastructures for High-Performance Computing: Cloud Infrastructures. *Encycl. Bioinforma. Comput. Biol.* 240–246 (2019).

406. Joshi, N. & Shah, S. A Comprehensive Survey of Services Provided by Prevalent Cloud Computing Environments. In *Smart Intelligent Computing and Applications*, 413–424 (Springer, 2019).

407. Gupta, A., Goswami, P., Chaudhary, N. & Bansal, R. Deploying an Application Using Google Cloud Platform. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 236–239 (IEEE, 2020).

408. Ooi, B. C. *et al.* SINGA: A Distributed Deep Learning Platform. In *Proceedings of the 23rd ACM international Conference on Multimedia*, 685–688 (2015).

409. Apache SINGA License. Online: https://github.com/apache/singa/blob/master/LICENSE (2020).

410. Dai, J. J. *et al.* BigDL: A Distributed Deep Learning Framework for Big Data. In *Proceedings of the ACM Symposium on Cloud Computing*, 50–60 (2019).

411. BigDL License. Online: https://github.com/intel-analytics/BigDL/blob/master/LICENSE (2020).

412. Jia, Y. *et al.* Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, 675–678 (2014).

413. Synced. Caffe2 Merges with PyTorch. Medium, Online: https://medium.com/@Synced/caffe2-merges-with-pytorch-a89c70ad9eb7 (2004).

414. Caffe License. Online: https://github.com/BVLC/caffe/blob/master/LICENSE (2017).

415. Tokui, S., Oono, K., Hido, S. & Clayton, J. Chainer: A Next-Generation Open Source Framework for Deep Learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Twenty-Ninth Annual Conference on Neural Information Processing Systems (NIPS)*, vol. 5, 1–6 (2015).

416. Chainer License. Online: https://docs.chainer.org/en/stable/license.html (2020).

417. Gibson, A. *et al.* Deeplearning4j: Distributed, Open-Source Deep Learning for Java and Scala on Hadoop and Spark. Towards Data Science, Online: https://deeplearning4j.org (2016).

418. Deeplearning4j License. Online: https://github.com/eclipse/deeplearning4j/blob/master/LICENSE (2020).

419. King, D. E. Dlib-ml: A Machine Learning Toolkit. *The J. Mach. Learn. Res.* **10**, 1755–1758 (2009).

420. Dlib C++ Library. Online: http://dlib.net (2020).

421. Dlib License. Online: https://github.com/davisking/dlib/blob/master/dlib/LICENSE.txt (2020).

422. Innes, M. Flux: Elegant Machine Learning with Julia. *J. Open Source Softw.* **3**, 602 (2018).

423. Flux License. Online: https://github.com/FluxML/Flux.jl/blob/master/LICENSE.md (2020).

424. Beale, M., Hagan, M. & Demuth, H. PDF Documentation: MATLAB Deep Learning Toolbox User's Guide. Online: https://uk.mathworks.com/help/deeplearning (2020).

425. MATLAB License. Online: https://mathworks.com/pricing-licensing.html (2020).

426. Seide, F. Keynote: The Computer Science Behind the Microsoft Cognitive Toolkit: An Open Source Large-Scale Deep Learning Toolkit for Windows and Linux. In *2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, xi–xi (IEEE, 2017).

427. CNTK License. Online: https://github.com/microsoft/CNTK/blob/master/LICENSE.md (2020).

428. Chen, T. *et al.* MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *arXiv preprint arXiv:1512.01274* (2015).

429. MXNet License. Online: https://github.com/apache/incubator-mxnet/blob/master/LICENSE (2020).

430. OpenNN. Online: https://www.opennn.net (2020).

431. OpenNN License. Online: https://github.com/Artelnics/OpenNN/blob/master/LICENSE.txt (2020).

432. Ma, Y., Yu, D., Wu, T. & Wang, H. PaddlePaddle: An Open-Source Deep Learning Platform from Industrial Practice. *Front. Data Comput.* **1**, 105–115 (2019).

433. PaddlePaddle License. Online: https://github.com/PaddlePaddle/Paddle/blob/develop/LICENSE (2020).

434. Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 8024–8035 (2019).

435. PyTorch License. Online: https://github.com/pytorch/pytorch/blob/master/LICENSE (2020).

436. Abadi, M. *et al.* TensorFlow: A System for Large-Scale Machine Learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283 (2016).

437. Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467* (2016).

438. TensorFlow License. Online: https://github.com/tensorflow/tensorflow/blob/master/LICENSE (2020).

439. Team, T. T. D. *et al.* Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv preprint arXiv:1605.02688* (2016).

440. Ketkar, N. Introduction to Theano. In *Deep Learning with Python*, 35–61 (Springer, 2017).

441. Theano License. Online: https://github.com/Theano/Theano/blob/master/doc/LICENSE.txt (2020).

442. Collobert, R., Bengio, S. & Mariéthoz, J. Torch: A Modular Machine Learning Software Library. Tech. Rep., Idiap (2002).

443. Torch License. Online: https://github.com/torch/torch7/blob/master/COPYRIGHT.txt (2020).

444. Mathematica Neural Networks Documentation. Online: https://reference.wolfram.com/language/guide/NeuralNetworks.html (2020).

445. Mathematica Licenses. Online: https://www.wolfram.com/legal (2020).

446. Li, M. *et al.* The Deep Learning Compiler: A Comprehensive Survey. *arXiv preprint arXiv:2002.03794* (2020).

447. Nguyen, G. *et al.* Machine Learning and Deep Learning Frameworks and Libraries for Large-Scale Data Mining: A Survey. *Artif. Intell. Rev.* **52**, 77–124 (2019).

448. Dai, W. & Berleant, D. Qualitative Benchmarking of Deep Learning Hardware and Frameworks: Review and Tutorial. *arXiv preprint arXiv:1907.03626* (2019).

449. Kharkovyna, O. Top 10 Best Deep Learning Frameworks in 2019. Towards Data Science, Online: https://towardsdatascience.com/top-10-best-deep-learning-frameworks-in-2019-5ccb90ea6de (2019).

450. Zacharias, J., Barz, M. & Sonntag, D. A Survey on Deep Learning Toolkits and Libraries for Intelligent User Interfaces. *arXiv preprint arXiv:1803.04818* (2018).

451. Parvat, A., Chavan, J., Kadam, S., Dev, S. & Pathak, V. A Survey of Deep-Learning Frameworks. In *2017 International Conference on Inventive Systems and Control (ICISC)*, 1–7 (IEEE, 2017).

452. Erickson, B. J., Korfiatis, P., Akkus, Z., Kline, T. & Philbrick, K. Toolkits and Libraries for Deep Learning. *J. Digit. Imaging* **30**, 400–405 (2017).

453. Baydin, A. G., Pearlmutter, B. A., Radul, A. A. & Siskind, J. M. Automatic Differentiation in Machine Learning: A Survey. *The J. Mach. Learn. Res.* **18**, 5595–5637 (2017).

454. Barham, P. & Isard, M. Machine Learning Systems are Stuck in a Rut. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, 177–183 (2019).

455. Afif, M., Said, Y. & Atri, M. Computer Vision Algorithms Acceleration Using Graphic Processors NVIDIA CUDA. *Clust. Comput.* 1–13 (2020).

456. Cook, S. *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2012), 1st edn.

457. Nickolls, J., Buck, I., Garland, M. & Skadron, K. Scalable Parallel Programming with CUDA. *Queue* **6**, 40–53 (2008).

458. Jordà, M., Valero-Lara, P. & Peña, A. J. Performance Evaluation of cuDNN Convolution Algorithms on NVIDIA Volta GPUs. *IEEE Access* **7**, 70461–70473 (2019).

459. de Supinski, B. R. *et al.* The Ongoing Evolution of OpenMP. *Proc. IEEE* **106**, 2004–2019 (2018).

460. Dagum, L. & Menon, R. OpenMP: An Industry Standard API for Shared-Memory Programming. *IEEE Comput. Sci. Eng.* **5**, 46–55 (1998).

461. He, H. The State of Machine Learning Frameworks in 2019. The Gradient, Online: https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry (2019).

462. Papers With Code: Trends. https://paperswithcode.com/trends (2020).

463. TensorFlow Libraries and Extensions. Online: https://www.tensorflow.org/resources/libraries-extensions (2020).

464. Chollet, F. *et al.* Keras. Online: https://keras.io (2020).

465. Sonnet repository. Online: https://github.com/deepmind/sonnet (2020).

466. Vaswani, A. *et al.* Tensor2tensor for Neural Machine Translation. *arXiv preprint arXiv:1803.07416* (2018).

467. Tang, Y. TF.Learn: TensorFlow's High-Level Module for Distributed Machine Learning. *arXiv preprint arXiv:1612.04251* (2016).

468. Damien, A. *et al.* TFLearn Repository. Online: https://github.com/tflearn/tflearn (2019).

469. TensorFlow Addons. Online: https://github.com/tensorflow/addons (2020).

470. Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, Eugene Brevdo. TF-Agents: A Library for Reinforcement Learning in TensorFlow. Online: https://github.com/tensorflow/agents (2018).

471. Castro, P. S., Moitra, S., Gelada, C., Kumar, S. & Bellemare, M. G. Dopamine: A Research Framework for Deep Reinforcement Learning. *arXiv preprint arXiv:1812.06110* (2018).

472. McMahan, B. & Ramage, D. Federated Learning: Collaborative Machine Learning Without Centralized Training Data. *Google Res. Blog* **3** (2017).

473. TensorFlow Federated. Online: https://github.com/tensorflow/federated (2018).

474. Caldas, S. *et al.* LEAF: A Benchmark for Federated Settings. *arXiv preprint arXiv:1812.01097* (2018).

475. Dillon, J. V. *et al.* TensorFlow Distributions. *arXiv preprint arXiv:1711.10604* (2017).

476. Hessel, M., Martic, M., de Las Casas, D. & Barth-Maron, G. Open Sourcing TRFL: A Library of Reinforcement Learning Building Blocks. DeepMind Blog, Online: https://blog.paperspace.com/geometric-deep-learning-framework-comparison (2018).

477. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

478. ANNdotNET. Online: https://github.com/bhrnjica/anndotnet (2020).

479. Create ML Documentation. Online: https://developer.apple.com/documentation/createml (2020).

480. Deep Cognition. Online: https://deepcognition.ai (2020).

481. MathWorks Deep Network Designer. Online: https://uk.mathworks.com/help/deeplearning/ref/deepnetworkdesigner-app.html (2020).

482. DIGITS. Online: https://developer.nvidia.com/digits (2020).

483. ENNUI. Online: https://math.mit.edu/ennui (2020).

484. Expresso. Online: http://val.serc.iisc.ernet.in/expresso (2020).

485. Neural Designer: Data Science and Machine Learning Platform. Online: https://www.neuraldesigner.com (2020).

486. Witten, I. H., Frank, E., Hall, M. A. & Pal, C. J. *Data Mining: Practical Machine Learning Tools and Techniques* (Morgan Kaufmann, 2016).

487. Hall, M. *et al.* The WEKA Data Mining Software: An Update. *ACM SIGKDD Explor. Newsl.* **11**, 10–18 (2009).

488. Holmes, G., Donkin, A. & Witten, I. H. WEKA: A Machine Learning Workbench. In *Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference*, 357–361 (IEEE, 1994).

489. Von Chamier, L. *et al.* ZeroCostDL4Mic: An Open Platform to Simplify Access and Use of Deep-Learning in Microscopy. *BioRxiv* (2020).

490. Ye, J. C. & Sung, W. K. Understanding Geometry of Encoder-Decoder CNNs. *arXiv preprint arXiv:1901.07647* (2019).

491. Ye, J. C., Han, Y. & Cha, E. Deep Convolutional Framelets: A General Deep Learning Framework for Inverse Problems. *SIAM J. on Imaging Sci.* **11**, 991–1048 (2018).

492. Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, 3104–3112 (2014).

493. List of Collections of Pretrained Models. Online: https://awesomeopensource.com/projects/pretrained-models (2020).

494. Model Zoo. Online: https://modelzoo.co (2020).

495. Open Neural Network Exchange. Online: https://onnx.ai (2020).

496. Bai, J., Lu, F., Zhang, K. *et al.* ONNX: Open Neural Network Exchange. Online: https://github.com/onnx/onnx (2020).

497. Shah, S. Microsoft and Facebook's Open AI Ecosystem Gains More Support. Engadget, Online: https://www.engadget.com/2017/10/11/microsoft-facebooks-ai-onxx-partners (2017).

498. Boyd, E. Microsoft and Facebook Create Open Ecosystem for AI Model Interoperability. Microsoft Azure Blog, Online: https://azure.microsoft.com/en-us/blog/microsoft-and-facebook-create-open-ecosystem-for-ai-model-interoperability (2017).

499. ONNX Model Zoo. Online: https://github.com/onnx/models (2020).

500. Gordon, J. Introducing TensorFlow Hub: A Library for Reusable Machine Learning Modules in TensorFlow. Medium, Online: https://tfhub.dev (2018).

501. TensorFlow Hub. Online: https://tfhub.dev (2020).

502. TensorFlow Model Garden. Online: https://github.com/tensorflow/models (2020).

503. Liang, H., Fu, W. & Yi, F. A Survey of Recent Advances in Transfer Learning. In *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, 1516–1523 (IEEE, 2019).

504. Zhuang, F. *et al.* A Comprehensive Survey on Transfer Learning. *arXiv preprint arXiv:1911.02685* (2019).

505. Tan, C. *et al.* A Survey on Deep Transfer Learning. In *International Conference on Artificial Neural Networks*, 270–279 (Springer, 2018).

506. Marcelino, P. Transfer Learning From Pre-Trained Models. Towards Data Science, Online: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751 (2018).

507. Weiss, K., Khoshgoftaar, T. M. & Wang, D. A Survey of Transfer Learning. *J. Big data* **3**, 9 (2016).

508. Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. How Transferable are Features in Deep Neural Networks? In *Advances in Neural Information Processing Systems*, 3320–3328 (2014).

509. Da Silva, F. L., Warnell, G., Costa, A. H. R. & Stone, P. Agents Teaching Agents: A Survey on Inter-Agent Transfer Learning. *Auton. Agents Multi-Agent Syst.* **34**, 9 (2020).

510. Shermin, T. *et al.* Enhanced Transfer Learning with ImageNet Trained Classification Layer. In *Pacific-Rim Symposium on Image and Video Technology*, 142–155 (Springer, 2019).

511. Ada, S. E., Ugur, E. & Akin, H. L. Generalization in Transfer Learning. *arXiv preprint arXiv:1909.01331* (2019).

512. The Khronos NNEF Working Group. Neural Network Exchange Format. Online: https://www.khronos.org/registry/NNEF (2020).

513. The HDF Group. Hierarchical Data Format, Version 5. Online: http://www.hdfgroup.org/HDF5 (2020).

514. HDF5 for Python. Online: http://www.h5py.org (2020).

515. Somnath, S., Smith, C. R., Laanait, N., Vasudevan, R. K. & Jesse, S. USID and Pycroscopy – Open Source Frameworks for Storing and Analyzing Imaging and Spectroscopy Data. *Microsc. Microanal.* **25**, 220–221 (2019).

516. Pycroscopy Repository. Online: https://github.com/pycroscopy/pycroscopy (2020).

517. HyperSpy. Online: https://hyperspy.org (2020).

518. de la Peña, F. *et al.* Electron Microscopy (Big and Small) Data Analysis with the Open Source Software Package HyperSpy. *Microsc. Microanal.* **23**, 214–215 (2017).

519. Rezk, N. M., Purnaprajna, M., Nordström, T. & Ul-Abdin, Z. Recurrent Neural Networks: An Embedded Computing Perspective. *IEEE Access* **8**, 57967–57996 (2020).

520. Du, K.-L. & Swamy, M. Recurrent Neural Networks. In *Neural Networks and Statistical Learning*, 351–371 (Springer, 2019).

521. Yu, Y., Si, X., Hu, C. & Zhang, J. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Comput.* **31**, 1235–1270 (2019).

522. Choe, Y. J., Shin, J. & Spencer, N. Probabilistic Interpretations of Recurrent Neural Networks. *Probabilistic Graph. Model.* (2017).

523. Choi, M., Kim, T. & Kim, J. Awesome Recurrent Neural Networks. Online: https://github.com/kjw0612/awesome-rnn (2017).

524. Lipton, Z. C., Berkowitz, J. & Elkan, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv preprint arXiv:1506.00019* (2015).

525. Hanin, B. & Rolnick, D. How to Start Training: The Effect of Initialization and Architecture. In *Advances in Neural Information Processing Systems*, 571–581 (2018).

526. Raschka, S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv preprint arXiv:1811.12808* (2018).

527. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258 (2017).

528. Everingham, M. *et al.* The PASCAL Visual Object Classes Challenge: A Retrospective. *Int. J. Comput. Vis.* **111**, 98–136 (2015).

529. Goyal, P. *et al.* Accurate, Large Minibatch SGD: Training Imagenet in 1 Hour. *arXiv preprint arXiv:1706.02677* (2017).

530. Laanait, N. *et al.* Exascale Deep Learning for Scientific Inverse Problems. *arXiv preprint arXiv:1909.11150* (2019).

531. Castelvecchi, D. Google Unveils Search Engine for Open Data. *Nature* **561**, 161–163 (2018).

532. Noy, N. Discovering Millions of Datasets on the Web. The Keyword, Online: https://blog.google/products/search/discovering-millions-datasets-web (2020).

533. Plesa, N. Machine Learning Datasets: A List of the Biggest Machine Learning Datasets From Across the Web. Online: https://www.datasetlist.com (2020).

534. Dua, D. & Graff, C. UCI Machine Learning Repository. Online: http://archive.ics.uci.edu/ml (2020).

535. Kaggle Datasets. Online: https://www.kaggle.com/datasets (2020).

536. VisualData. Online: https://www.visualdata.io/discovery (2020).

537. Vanschoren, J., Van Rijn, J. N., Bischl, B. & Torgo, L. OpenML: Networked Science in Machine Learning. *ACM SIGKDD Explor. Newsl.* **15**, 49–60 (2014).

538. Stanford, S. The Best Public Datasets for Machine Learning and Data Science. Towards AI, Online: https://towardsai.net/datasets (2020).

539. Datasets for Data Science and Machine Learning. Elite Data Science, Online: https://elitedatascience.com/datasets (2020).

540. Iderhoff, N. Natural Language Processing Datasets. Online: https://github.com/niderhoff/nlp-datasets (2020).

541. Deep Learning Datasets. Online: http://deeplearning.net/datasets (2017).

542. Hughes, I. & Hase, T. *Measurements and Their Uncertainties: A Practical Guide to Modern Error Analysis* (Oxford University Press, 2010).

543. Working Group 1 of the Joint Committee for Guides in Metrology. JCGM 100: 2008 Evaluation of Measurement Data – Guide to the Expression of Uncertainty in Measurement. International Bureau of Weights and Measures, Online: https://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf (2008).

544. Vaux, D. L., Fidler, F. & Cumming, G. Replicates and Repeats - What is the Difference and is it Significant? A Brief Discussion of Statistics and Experimental Design. *EMBO Reports* **13**, 291–296 (2012).

545. Urbach, P. On the Utility of Repeating the 'Same' Experiment. *Australas. J. Philos.* **59**, 151–162 (1981).

546. Musgrave, A. Popper and 'Diminishing Returns From Repeated Tests'. *Australas. J. Philos.* **53**, 248–253 (1975).

547. Senior, A. W. *et al.* Improved Protein Structure Prediction Using Potentials From Deep Learning. *Nature* **577**, 706–710 (2020).

548. Voß, H., Heck, C. A., Schallmey, M. & Schallmey, A. Database Mining for Novel Bacterial β-Etherases, Glutathione-Dependent Lignin-Degrading Enzymes. *Appl. Environ. Microbiol.* **86** (2020).

549. Papers With Code State-of-the-Art Leaderboards. Online: https://paperswithcode.com/sota (2020).

550. Krizhevsky, A., Nair, V. & Hinton, G. The CIFAR-10 Dataset. Online: http://www.cs.toronto.edu/~kriz/cifar.html (2014).

551. Krizhevsky, A. & Hinton, G. Learning Multiple Layers of Features from Tiny Images. Tech. Rep., Citeseer (2009).

552. LeCun, Y., Cortes, C. & Burges, C. MNIST Handwritten Digit Database. AT&T Labs, Online: http://yann.lecun.com/exdb/mnist (2010).

553. Russakovsky, O. *et al.* ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **115**, 211–252 (2015).

554. Open Access Directory Data Repositories. Online: http://oad.simmons.edu/oadwiki/Data_repositories (2020).

555. Nature Scientific Data Rrecommended Data Repositories. Online: https://www.nature.com/sdata/policies/repositories (2020).

556. Zenodo. Online: https://about.zenodo.org (2020).

557. Zenodo Frequently Asked Questions. Online: https://help.zenodo.org (2020).

558. Ortega, D. R. *et al.* ETDB-Caltech: A Blockchain-Based Distributed Public Database for Electron Tomography. *PLOS ONE* **14**, e0215531 (2019).

559. EMDataResource: Unified Data Resource for 3DEM. Online: https://www.emdataresource.org/index.html (2020).

560. Lawson, C. L. *et al.* EMDataBank Unified Data Resource for 3DEM. *Nucleic Acids Res.* **44**, D396–D403 (2016).

561. Esquivel-Rodríguez, J. *et al.* Navigating 3D Electron microscopy Maps with EM-SURFER. *BMC Bioinforma.* **16**, 181 (2015).

562. Lawson, C. L. *et al.* EMDataBank.org: Unified Data Resource for CryoEM. *Nucleic Acids Res.* **39**, D456–D464 (2010).

563. Henrick, K., Newman, R., Tagari, M. & Chagoyen, M. EMDep: A Web-Based System for the Deposition and Validation of High-Resolution Electron Microscopy Macromolecular Structural Information. *J. Struct. Biol.* **144**, 228–237 (2003).

564. Tagari, M., Newman, R., Chagoyen, M., Carazo, J.-M. & Henrick, K. New Electron Microscopy Database and Deposition System. *Trends Biochem. Sci.* **27**, 589 (2002).

565. Iudin, A., Korir, P. K., Salavert-Torres, J., Kleywegt, G. J. & Patwardhan, A. EMPIAR: A Public Archive for Raw Electron Microscopy Image Data. *Nat. Methods* **13**, 387 (2016).

566. Hey, T., Butler, K., Jackson, S. & Thiyagalingam, J. Machine Learning and Big Scientific Data. *Philos. Transactions Royal Soc. A* **378**, 20190054 (2020).

567. Aversa, R., Modarres, M. H., Cozzini, S., Ciancio, R. & Chiusole, A. The First Annotated Set of Scanning Electron Microscopy Images for Nanoscience. *Sci. Data* **5**, 180172 (2018).

568. Levin, B. D. *et al.* Nanomaterial Datasets to Advance Tomography in Scanning Transmission Electron Microscopy. *Sci. Data* **3**, 1–11 (2016).

569. Cerius^2 Modeling Environment: File Formats. Online: http://www.chem.cmu.edu/courses/09-560/docs/msi/modenv/D_Files.html (2020).

570. CrystalMaker: File Formats Supported. Online: http://www.crystalmaker.com/support/advice/index.html?topic=cm-file-formats (2020).

571. Bernstein, H. J. *et al.* Specification of the Crystallographic Information File format, Version 2.0. *J. Appl. Crystallogr.* **49**, 277–284 (2016).

572. Hall, S. R. & McMahon, B. The Implementation and Evolution of STAR/CIF Ontologies: Interoperability and Preservation of Structured Data. *Data Sci. J.* **15** (2016).

573. Brown, I. D. & McMahon, B. CIF: The Computer Language of Crystallography. *Acta Crystallogr. Sect. B: Struct. Sci.* **58**, 317–324 (2002).

574. Hall, S. R., Allen, F. H. & Brown, I. D. The Crystallographic Information File (CIF): A New Standard Archive File for Crystallography. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **47**, 655–685 (1991).

575. Bruno, I. *et al.* Crystallography and Databases. *Data Sci. J.* **16** (2017).

576. Crystallographic Databases and Related Resources. Online: https://www.iucr.org/resources/data/databases (2020).

577. Crystal Structure Databases. Online: https://serc.carleton.edu/research_education/crystallography/xldatabases.html (2020).

578. Quirós, M., Gražulis, S., Girdzijauskaitė, S., Merkys, A. & Vaitkus, A. Using SMILES Strings for the Description of Chemical Connectivity in the Crystallography Open Database. *J. Cheminformatics* **10**, DOI: 10.1186/s13321-018-0279-6 (2018).

579. Merkys, A. *et al.* COD::CIF::Parser: An Error-Correcting CIF Parser for the Perl Language. *J. Appl. Crystallogr.* **49**, DOI: 10.1107/S1600576715022396 (2016).

580. Gražulis, S., Merkys, A., Vaitkus, A. & Okulič-Kazarinas, M. Computing Stoichiometric Molecular Composition From Crystal Structures. *J. Appl. Crystallogr.* **48**, 85–91, DOI: 10.1107/S1600576714025904 (2015).

581. Gražulis, S. *et al.* Crystallography Open Database (COD): An Open-Access Collection of Crystal Structures and Platform for World-Wide Collaboration. *Nucleic Acids Res.* **40**, D420–D427, DOI: 10.1093/nar/gkr900 (2012). http://nar.oxfordjournals.org/content/40/D1/D420.full.pdf+html.

582. Gražulis, S. *et al.* Crystallography Open Database – An Open-Access Collection of Crystal Structures. *J. Appl. Crystallogr.* **42**, 726–729, DOI: 10.1107/S0021889809016690 (2009).

583. Downs, R. T. & Hall-Wallace, M. The American Mineralogist Crystal Structure Database. *Am. Mineral.* **88**, 247–250 (2003).

584. Zagorac, D., Müller, H., Ruehl, S., Zagorac, J. & Rehme, S. Recent Developments in the Inorganic Crystal Structure Database: Theoretical Crystal Structure Data and Related Features. *J. Appl. Crystallogr.* **52** (2019).

585. Allmann, R. & Hinek, R. The Introduction of Structure Types into the Inorganic Crystal Structure Database ICSD. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **63**, 412–417 (2007).

586. Hellenbrandt, M. The Inorganic Crystal Structure Database (ICSD) - Present and Future. *Crystallogr. Rev.* **10**, 17–22 (2004).

587. Belsky, A., Hellenbrandt, M., Karen, V. L. & Luksch, P. New Developments in the Inorganic Crystal Structure Database (ICSD): Accessibility in Support of Materials Research and Design. *Acta Crystallogr. Sect. B: Struct. Sci.* **58**, 364–369 (2002).

588. Bergerhoff, G., Brown, I., Allen, F. *et al.* Crystallographic Databases. *Int. Union Crystallogr. Chester* **360**, 77–95 (1987).

589. Mighell, A. D. & Karen, V. L. NIST Crystallographic Databases for Research and Analysis. *J. Res. Natl. Inst. Standards Technol.* **101**, 273 (1996).

590. NIST Standard Reference Database 3. Online: https://www.nist.gov/srd/nist-standard-reference-database-3 (2020).

591. Kay, W. *et al.* The Kinetics Human Action Video Dataset. *arXiv preprint arXiv:1705.06950* (2017).

592. Abu-El-Haija, S. *et al.* YouTube-8M: A Large-Scale Video Classification Benchmark. *arXiv preprint arXiv:1609.08675* (2016).

593. Rehm, G. *et al.* QURATOR: Innovative Technologies for Content and Data Curation. *arXiv preprint arXiv:2004.12195* (2020).

594. van der Voort, S. R., Smits, M. & Klein, S. DeepDicomSort: An Automatic Sorting Algorithm for Brain Magnetic Resonance Imaging Data. *Neuroinformatics* 1–26 (2020).

595. Pezoulas, V. C. *et al.* Medical Data Quality Assessment: On the Development of an Automated Framework for Medical Data Curation. *Comput. Biol. Medicine* **107**, 270–283 (2019).

596. Bhat, M. *et al.* ADeX: A Tool for Automatic Curation of Design Decision Knowledge for Architectural Decision recommendations. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, 158–161 (IEEE, 2019).

597. Thirumuruganathan, S., Tang, N., Ouzzani, M. & Doan, A. Data curation with deep learning [vision]. *arXiv preprint arXiv:1803.01384* (2018).

598. Lee, K. *et al.* Scaling up Data Curation Using Deep Learning: An application to Literature Triage in Genomic Variation Resources. *PLoS Comput. Biol.* **14**, e1006390 (2018).

599. Freitas, A. & Curry, E. Big Data Curation. In *New Horizons for a Data-Driven Economy*, 87–118 (Springer, 2016).

600. European Microcredit Whitepaper. Online: https://www.european-microfinance.org/sites/default/files/document/file/paris_europlace_whitepaper_on_microfinance_july_2019.pdf (2019).

601. Di Cosmo, R. & Zacchiroli, S. Software Heritage: Why and How to Preserve Software Source Code. In *Proceedings of 14th International Conference on Digital Preservation (iPRES2017)* (2017).

602. Apache Allura. Online: https://allura.apache.org (2020).

603. AWS CodeCommit. Online: https://aws.amazon.com/codecommit (2020).

604. Beanstalk. Online: https://beanstalkapp.com (2020).

605. BitBucket. Online: https://bitbucket.org/product (2020).

606. GitHub. Online: https://github.com (2020).

607. GitLab. Online: https://about.gitlab.com (2020).

608. Gogs. Online: https://gogs.io (2020).

609. Google Cloud Source Repositories. Online: https://cloud.google.com/source-repositories (2020).

610. Launchpad. Online: https://launchpad.net (2020).

611. Phabricator. Online: https://www.phacility.com/phabricator (2020).

612. Savannah. Online: https://savannah.gnu.org (2020).

613. SourceForge. Online: https://sourceforge.net (2020).

614. Sheoran, J., Blincoe, K., Kalliamvakou, E., Damian, D. & Ell, J. Understanding Watchers on GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, 336–339 (2014).

615. Vale, G., Schmid, A., Santos, A. R., De Almeida, E. S. & Apel, S. On the Relation Between GitHub Communication Activity and Merge Conflicts. *Empir. Softw. Eng.* **25**, 402–433 (2020).

616. Bao, L., Xia, X., Lo, D. & Murphy, G. C. A Large Scale Study of Long-Time Contributor Prediction for GitHub Projects. *IEEE Transactions on Softw. Eng.* (2019).

617. Elazhary, O., Storey, M.-A., Ernst, N. & Zaidman, A. Do as I Do, Not as I Say: Do Contribution Guidelines Match the GitHub Contribution Process? In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 286–290 (IEEE, 2019).

618. Pinto, G., Steinmacher, I. & Gerosa, M. A. More Common than Tou Think: An In-Depth Study of Casual Contributors. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, 112–123 (IEEE, 2016).

619. Kobayakawa, N. & Yoshida, K. How GitHub Contributing.md Contributes to Contributors. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 694–696 (IEEE, 2017).

620. Lu, Y. *et al.* Studying in the 'Bazaar': An Exploratory Study of Crowdsourced Learning in GitHub. *IEEE Access* **7**, 58930–58944 (2019).

621. Qiu, H. S., Li, Y. L., Padala, S., Sarma, A. & Vasilescu, B. The Signals that Potential Contributors Look for When Choosing Open-source Projects. *Proc. ACM on Human-Computer Interact.* **3**, 1–29 (2019).

622. Alamer, G. & Alyahya, S. Open Source Software Hosting Platforms: A Collaborative Perspective's Review. *J. Softw.* **12**, 274–291 (2017).

623. Wikipedia Contributors. Comparison of source-code-hosting facilities — Wikipedia, the free encyclopedia. Online: https://en.wikipedia.org/w/index.php?title=Comparison_of_source-code-hosting_facilities&oldid=964020832 (2020). [Accessed 25-June-2020].

624. Apache Allura Feature Comparison. Online: https://forge-allura.apache.org/p/allura/wiki/Feature%20Comparison (2020).

625. Alexa Top Sites. Online: https://www.alexa.com/topsites (2020).

626. How are Alexa's Traffic Rankings Determined. Online: https://support.alexa.com/hc/en-us/articles/200449744-How-are-Alexa-s-traffic-rankings-determined- (2020).

627. Haider, J. & Sundin, O. *Invisible Search and Online Search Engines: The Ubiquity of Search in Everyday Life* (Routledge, 2019).

628. Vincent, N., Johnson, I., Sheehan, P. & Hecht, B. Measuring the Importance of User-Generated Content to Search Engines. In *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 13, 505–516 (2019).

629. Jain, A. The Role and Importance of Search Engine and Search Engine Optimization. *Int. J. Emerg. Trends & technology Comput. Sci.* **2**, 99–102 (2013).

630. Brin, S. & Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Comput. Networks* **30**, 107–117 (1998).

631. Fröbe, M., Bittner, J. P., Potthast, M. & Hagen, M. The effect of content-equivalent near-duplicates on the evaluation of search engines. In *European Conference on Information Retrieval*, 12–19 (Springer, 2020).

632. Kostagiolas, P., Strzelecki, A., Banou, C. & Lavranos, C. The Impact of Google on Discovering Scholarly Information: Managing STM publishers' Visibility in Google. *Collect. Curation* (2020).

633. Gul, S., Ali, S. & Hussain, A. Retrieval Performance of Google, Yahoo and Bing for Navigational Queries in the Field of "Life Science and Biomedicine". *Data Technol. Appl.* (2020).

634. Shafi, S. & Ali, S. Retrieval Performance of Select Search Engines in the Field of Physical Sciences. *NISCAIR-CSIR* 117–122 (2019).

635. Steiner, M., Magin, M., Stark, B. & Geiß, S. Seek and You Shall Find? A Content Analysis on the Diversity of Five Search Engines' Results on Political Queries. *Information, Commun. & Soc.* 1–25 (2020).

636. Wu, S., Zhang, Z. & Xu, C. Evaluating the Effectiveness of Web Search Engines on Results Diversification. *Inf. Res. An Int. Electron. J.* **24**, n1 (2019).

637. Rahim, I., Mushtaq, H., Ahmad, S. *et al.* Evaluation of Search Engines Using Advanced Search: Comparative Analysis of Yahoo and Bing. *Libr. Philos. Pract.* 1–12 (2019).

638. Tazehkandi, M. Z. & Nowkarizi, M. Evaluating the Effectiveness of Google, Parsijoo, Rismoon, and Yooz to Retrieve Persian Documents. *Libr. Hi Tech* (2020).

639. Gusenbauer, M. Google Scholar to Overshadow Them All? Comparing the Sizes of 12 Academic Search Engines and Bibliographic Databases. *Scientometrics* **118**, 177–214 (2019).

640. Hook, D. W., Porter, S. J. & Herzog, C. Dimensions: Building Context for Search and Evaluation. *Front. Res. Metrics Anal.* **3**, 23 (2018).

641. Bates, J., Best, P., McQuilkin, J. & Taylor, B. Will Web Search Engines Replace Bibliographic Databases in the Systematic Identification of Research? *The J. Acad. Librariansh.* **43**, 8–17 (2017).

642. Verheggen, K. *et al.* Anatomy and Evolution of Database Search Engines – A Central Component of Mass Spectrometry Based Proteomic Workflows. *Mass Spectrom. Rev.* **39**, 292–306 (2020).

643. Li, S. *et al.* Deep Job Understanding at LinkedIn. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2145–2148 (2020).

644. Agazzi, A. E. Study of the Usability of LinkedIn: A Social Media Platform Meant to Connect Employers and Employees. *arXiv preprint arXiv:2006.03931* (2020).

645. Forrester, A., Björk, B.-C. & Tenopir, C. New Web Services that Help Authors Choose Journals. *Learn. Publ.* **30**, 281–287 (2017).

646. Kang, D. M., Lee, C. C., Lee, S. & Lee, W. Patent Prior Art Search Using Deep Learning Language Model. In *Proceedings of the 24th Symposium on International Database Engineering & Applications*, 1–5 (2020).

647. Kang, M., Lee, S. & Lee, W. Prior Art Search Using Multi-modal Embedding of Patent Documents. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 548–550 (IEEE, 2020).

648. Shalaby, W. & Zadrozny, W. Patent Retrieval: A Literature Review. *Knowl. Inf. Syst.* 1–30 (2019).

649. Khode, A. & Jambhorkar, S. A Literature Review on Patent Information Retrieval Techniques. *Indian J. Sci. Technol.* **10**, 1–13 (2017).

650. Kong, X., Shi, Y., Yu, S., Liu, J. & Xia, F. Academic Social Networks: Modeling, Analysis, Mining and Applications. *J. Netw. Comput. Appl.* **132**, 86–103 (2019).

651. Makri, K., Papadas, K. & Schlegelmilch, B. B. Global Social Networking Sites and Global Identity: A Three-Country Study. *J. Bus. Res.* (2019).

652. Acquisti, A. & Fong, C. An Experiment in Hiring Discrimination via Online Social Networks. *Manag. Sci.* **66**, 1005–1024 (2020).

653. Mustafaraj, E., Lurie, E. & Devine, C. The Case for Voter-Centered Audits of Search Engines During Political Elections. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 559–569 (2020).

654. Kulshrestha, J. *et al.* Search Bias Quantification: Investigating Political Bias in Social Media and Web Search. *Inf. Retr. J.* **22**, 188–227 (2019).

655. Puschmann, C. Beyond the Bubble: Assessing the Diversity of Political Search Results. *Digit. Journalism* **7**, 824–843 (2019).

656. Ray, L. 2020 Google Search Survey: How Much Do Users Trust Their Search Results? MOZ, Online: https://moz.com/blog/2020-google-search-survey (2020).

657. Johnson, D. M. Lectures, Textbooks, Academic Calendar, and Administration: An Agenda for Change. In *The Uncertain Future of American Public Higher Education*, 75–89 (Springer, 2019).

658. Lin, H. Teaching and Learning Without a Textbook: Undergraduate Student Perceptions of Open Educational Resources. *Int. Rev. Res. Open Distributed Learn.* **20** (2019).

659. Stack Overflow. Online: https://stackoverflow.com/tour (2020).

660. Wu, Y., Wang, S., Bezemer, C.-P. & Inoue, K. How do Developers Utilize Source Code from Stack Overflow? *Empir. Softw. Eng.* **24**, 637–673 (2019).

661. Zhang, H., Wang, S., Chen, T.-H. & Hassan, A. E. Reading Answers on Stack Overflow: Not Enough! *IEEE Transactions on Softw. Eng.* (2019).

662. Zhang, T., Gao, C., Ma, L., Lyu, M. & Kim, M. An Empirical Study of Common Challenges in Developing Deep Learning Applications. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 104–115 (IEEE, 2019).

663. Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G. & Oliveto, R. Toxic Code Snippets on Stack Overflow. *IEEE Transactions on Softw. Eng.* (2019).

664. Zhang, T., Upadhyaya, G., Reinhardt, A., Rajan, H. & Kim, M. Are Code Examples on an Online Q&A Forum Reliable?: A Study of API Misuse on Stack Overflow. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 886–896 (IEEE, 2018).

665. Medium. Online: https://medium.com (2020).

666. Machine Learning Subreddit. Reddit, Online: https://www.reddit.com/r/MachineLearning (2020).

667. Learn Machine Learning Subreddit. Reddit, Online: https://www.reddit.com/r/learnmachinelearning (2020).

668. Mitchell, D. R. G. & Schaffer, B. Scripting-Customised Microscopy Tools for Digital Micrograph. *Ultramicroscopy* **103**, 319–332 (2005).

669. DigitalMicrograph Scripts. Online: http://www.dmscripting.com/scripts.html (2020).

670. Internet Archive. Online: ttps://archive.org (2020).

671. Kanhabua, N. *et al.* How to Search the Internet Archive Without Indexing It. In *International Conference on Theory and Practice of Digital Libraries*, 147–160 (Springer, 2016).

672. Internet Archive Wayback Machine. Online: https://archive.org/web (2020).

673. Bowyer, S. The Wayback Machine: Notes on a Re-Enchantment. *Arch. Sci.* 1–15 (2020).

674. Grotke, A. Web Archiving at the Library of Congress. *Comput. Libr.* **31**, 15–19 (2011).

675. About Distill. Online: https://distill.pub/about (2020).

676. Lewinson, E. My 10 Favorite Resources for Learning Data Science Online. Towards Data Science, Online: https://towardsdatascience.com/my-10-favorite-resources-for-learning-data-science-online-c645aa3d0afb (2020).

677. Chadha, H. S. Handpicked Resources for Learning Deep Learning in 2020. Towards Data Science, Online: https://towardsdatascience.com/handpicked-resources-for-learning-deep-learning-in-2020-e50c6768ab6e (2020).

678. Besbes, A. Here Are My Top Resources to Learn Deep Learning. Towards Data Science, Online: https://medium.com/datadriveninvestor/my-top-resources-to-learn-deep-learning-a14d1fc8e95a (2020).

679. Hutson, M. Artificial Intelligence Faces Reproducibility Crisis (2018).

680. Baker, M. Reproducibility Crisis? *Nature* **533**, 353–66 (2016).

681. Sethi, A., Sankaran, A., Panwar, N., Khare, S. & Mani, S. DLPaper2Code: Auto-Generation of Code from Deep Learning Research Papers. *arXiv preprint arXiv:1711.03543* (2017).

682. Tan, Z.-Y., Cai, N., Zhou, J. & Zhang, S.-G. On Performance of Peer Review for Academic Journals: Analysis Based on Distributed Parallel System. *IEEE Access* **7**, 19024–19032 (2019).

683. Kim, L., Portenoy, J. H., West, J. D. & Stovel, K. W. Scientific Journals Still Matter in the Era of Academic Search Engines and Preprint Archives. *J. Assoc. for Inf. Sci. Technol.* (2019).

684. Rallison, S. What are Journals For? *The Annals The Royal Coll. Surg. Engl.* **97**, 89–91 (2015).

685. Bornmann, L. & Mutz, R. Growth Rates of Modern Science: A Bibliometric Analysis Based on the Number of Publications and Cited References. *J. Assoc. for Inf. Sci. Technol.* **66**, 2215–2222 (2015).

686. Kaldas, M., Michael, S., Hanna, J. & Yousef, G. M. Journal Impact Factor: A Bumpy Ride in an Open Space. *J. Investig. Medicine* **68**, 83–87 (2020).

687. Orbay, K., Miranda, R. & Orbay, M. Building Journal Impact Factor Quartile into the Assessment of Academic Performance: A Case Study. *arXiv preprint arXiv:2005.02726* (2020).

688. Lei, L. & Sun, Y. Should Highly Cited Items be Excluded in Impact Factor Calculation? The Effect of Review Articles on Journal Impact Factor. *Scientometrics* **122**, 1697–1706 (2020).

689. Top Most Research Tools For Selecting The Best Journal For Your Research Article. Pubrica, https://pubrica.com/academy/2019/11/14/topmost-research-tools-for-selecting-the-best-journal-for-your-research-article (2019).

690. Hoy, M. B. Rise of the Rxivs: How Preprint Servers are Changing the Publishing Process. *Med. Ref. Serv. Q.* **39**, 84–89 (2020).

691. Fry, N. K., Marshall, H. & Mellins-Cohen, T. In Praise of Preprints. *Microb. Genomics* **5** (2019).

692. Rodríguez, E. G. Preprints and Preprint Servers as Academic Communication Tools. *Revista Cuba. de Información en Ciencias de la Salud* **30** (2019).

693. Ginsparg, P. ArXiv at 20. *Nature* **476**, 145–147 (2011).

694. Furnival, A. C. & Hubbard, B. Open Access to Scholarly Communications: Advantages, Policy and Advocacy. *Acceso Abierto a la información en las Bibliotecas Académicas de América Latina y el Caribe* 101–120 (2020).

695. Niyazov, Y. *et al.* Open Access Meets Discoverability: Citations to Articles Posted to Academia.edu. *PLOS ONE* **11**, e0148257 (2016).

696. Robinson-Garcia, N., Costas, R. & van Leeuwen, T. N. State of Open Access Penetration in Universities Worldwide. *arXiv preprint arXiv:2003.12273* (2020).

697. Siler, K. & Frenken, K. The Pricing of Open Access Journals: Diverse Niches and Sources of Value in Academic Publishing. *Quant. Sci. Stud.* **1**, 28–59 (2020).

698. Green, T. Is Open Access Affordable? Why Current Models Do Not Work and Why We Need Internet-Era Transformation of Scholarly Communications. *Learn. Publ.* **32**, 13–25 (2019).

699. Gadd, E., Fry, J. & Creaser, C. The Influence of Journal Publisher Characteristics on Open Access Policy Trends. *Scientometrics* **115**, 1371–1393 (2018).

700. Why Should You Publish in Machine Learning: Science and Technology? IOP Science, Online: https://iopscience.iop.org/journal/2632-2153/page/about-the-journal (2020).

701. Gibney, E. Open Journals that Piggyback on arXiv Gather Momentum. *Nat. News* **530**, 117 (2016).

702. Martínez-López, J. I., Barrón-González, S. & Martínez López, A. Which Are the Tools Available for Scholars? A Review of Assisting Software for Authors During Peer Reviewing Process. *Publications* **7**, 59 (2019).

703. Microsoft Word. Online: https://www.microsoft.com/en-gb/microsoft-365/word (2020).

704. 10 Free MS Word Alternatives You Can Use Today. Investintech, https://www.investintech.com/resources/articles/tenwordalternatives (2020).

705. Pignalberi, G. & Dominici, M. Introduction to LATEX and to Some of its Tools. *ArsTEXnica* 8 (2019).

706. Bransen, M. & Schulpen, G. Pimp your thesis: a minimal introduction to latex. IC/TC, U.S.S. Proton, Online: https://ussproton.nl/files/careerweeks/20180320-pimpyourthesis.pdf (2018).

707. Lamport, L. *LATEX: A document Preparation System: User's Guide and Reference Manual* (Addison-Wesley, 1994).

708. Matthews, D. Craft Beautiful Equations in Word with LaTeX (2019).

709. Knauff, M. & Nejasmic, J. An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development. *PloS one* **9**, e115069 (2014).

710. Why I Write with LaTeX (and Why You Should Too). Medium, Online: https://medium.com/@marko_kovic/why-i-write-with-latex-and-why-you-should-too-ba6a764fadf9 (2017).

711. Allington, D. The LaTeX Fetish (Or: Don't Write in LaTeX! It's Just for Typesetting). Online: http://www.danielallington.net/2016/09/the-latex-fetish (2016).

712. Overleaf Documentation. Online: https://www.overleaf.com/learn (2020).

713. Venkateshaiah, A. *et al.* Microscopic Techniques for the Analysis of Micro and Nanostructures of Biopolymers and Their Derivatives. *Polymers* **12**, 512 (2020).

714. Alqaheem, Y. & Alomair, A. A. Microscopy and Spectroscopy Techniques for Characterization of Polymeric Membranes. *Membranes* **10**, 33 (2020).

715. Morrison, K. *Characterisation Methods in Solid State and Materials Science* (IOP Publishing, 2019).

716. Maghsoudy-Louyeh, S., Kropf, M. & Tittmann, B. Review of Progress in Atomic Force Microscopy. *The Open Neuroimaging J.* **12** (2018).

717. Rugar, D. & Hansma, P. Atomic Force Microscopy. *Phys. Today* **43**, 23–30 (1990).

718. Krull, A., Hirsch, P., Rother, C., Schiffrin, A. & Krull, C. Artificial-Intelligence-Driven Scanning Probe Microscopy. *Commun. Phys.* **3**, 1–8 (2020).

719. Dutta, A. Fourier Transform Infrared Spectroscopy. In *Spectroscopic Methods for Nanomaterials Characterization*, 73–93 (Elsevier, 2017).

720. Griffiths, P. R. & De Haseth, J. A. *Fourier Transform Infrared Spectrometry*, vol. 171 (John Wiley & Sons, 2007).

721. Chien, P.-H., Griffith, K. J., Liu, H., Gan, Z. & Hu, Y.-Y. Recent Advances in Solid-State Nuclear Magnetic Resonance Techniques for Materials Research. *Annu. Rev. Mater. Res.* **50** (2020).

722. Lambert, J. B., Mazzola, E. P. & Ridge, C. D. *Nuclear Magnetic Resonance Spectroscopy: An Introduction to Principles, Applications, and Experimental Methods* (John Wiley & Sons, 2019).

723. Mlynárik, V. Introduction to Nuclear Magnetic Resonance. *Anal. Biochem.* **529**, 4–9 (2017).

724. Rabi, I. I., Zacharias, J. R., Millman, S. & Kusch, P. A New Method of Measuring Nuclear Magnetic Moment. *Phys. Rev.* **53**, 318 (1938).

725. Smith, E. & Dent, G. *Modern Raman Spectroscopy: A Practical Approach* (John Wiley & Sons, 2019).

726. Jones, R. R., Hooper, D. C., Zhang, L., Wolverson, D. & Valev, V. K. Raman techniques: Fundamentals and frontiers. *Nanoscale Res. Lett.* **14**, 1–34 (2019).

727. Ameh, E. A Review of Basic Crystallography and X-Ray Diffraction Applications. *The Int. J. Adv. Manuf. Technol.* **105**, 3289–3302 (2019).

728. Rostron, P., Gaber, S. & Gaber, D. Raman Spectroscopy, Review. *Int. J. Eng. Tech. Res.* **6**, 2454–4698 (2016).

729. Zhang, X., Tan, Q.-H., Wu, J.-B., Shi, W. & Tan, P.-H. Review on the Raman Spectroscopy of Different Types of Layered Materials. *Nanoscale* **8**, 6435–6450 (2016).

730. Epp, J. X-Ray Diffraction (XRD) Techniques for Materials Characterization. In *Materials Characterization Using Nondestructive Evaluation (NDE) Methods*, 81–124 (Elsevier, 2016).

731. Keren, S. *et al.* Noninvasive Molecular Imaging of Small Living Subjects using Raman Spectroscopy. *Proc. Natl. Acad. Sci.* **105**, 5844–5849 (2008).

732. Khan, H. *et al.* Experimental Methods in Chemical Engineering: X-Ray Diffraction Spectroscopy – XRD. *The Can. J. Chem. Eng.* **98**, 1255–1266 (2020).

733. Scarborough, N. M. *et al.* Dynamic X-Ray Diffraction Sampling for Protein Crystal Positioning. *J. Synchrotron Radiat.* **24**, 188–195 (2017).

734. Leani, J. J., Robledo, J. I. & Sánchez, H. J. Energy Dispersive Inelastic X-Ray Scattering Spectroscopy – A Review. *Spectrochimica Acta Part B: At. Spectrosc.* **154**, 10–24 (2019).

735. Vanhoof, C., Bacon, J. R., Fittschen, U. E. & Vincze, L. 2020 Atomic Spectrometry Update – A Review of Advances in X-Ray Fluorescence Spectrometry and its Special Applications. *J. Anal. At. Spectrom.* (2020).

736. Shackley, M. S. X-Ray Fluorescence Spectrometry (XRF). *The Encycl. Archaeol. Sci.* 1–5 (2018).

737. Greczynski, G. & Hultman, L. X-Ray Photoelectron Spectroscopy: Towards Reliable Binding Energy Referencing. *Prog. Mater. Sci.* **107**, 100591 (2020).

738. Baer, D. R. *et al.* Practical Guides for X-Ray Photoelectron Spectroscopy: First Steps in Planning, Conducting, and Reporting XPS Measurements. *J. Vac. Sci. & Technol. A: Vacuum, Surfaces, Films* **37**, 031401 (2019).

739. Du, M. & Jacobsen, C. Relative Merits and Limiting Factors for X-Ray and Electron Microscopy of Thick, Hydrated Organic Materials (Revised) (2020).

740. Hsu, T. Technique of Reflection Electron Microscopy. *Microsc. Res. Tech.* **20**, 318–332 (1992).

741. Yagi, K. Reflection Electron Microscopy. *J. Appl. Crystallogr.* **20**, 147–160 (1987).

742. Mohammed, A. & Abdullah, A. Scanning Electron Microscopy (SEM): A Review. In *Proceedings of the 2018 International Conference on Hydraulics and Pneumatics, Băile Govora, Romania*, 7–9 (2018).

743. Goldstein, J. I. *et al. Scanning Electron Microscopy and X-Ray Microanalysis* (Springer, 2017).

744. Keyse, R. *Introduction to Scanning Transmission Electron Microscopy* (Routledge, 2018).

745. Pennycook, S. J. & Nellist, P. D. *Scanning Transmission Electron Microscopy: Imaging and Analysis* (Springer Science & Business Media, 2011).

746. Sutter, P. Scanning Tunneling Microscopy in Surface Science. In *Springer Handbook of Microscopy*, 2–2 (Springer, 2019).

747. Voigtländer, B. *et al.* Invited Review Article: Multi-Tip Scanning Tunneling Microscopy: Experimental Techniques and Data Analysis. *Rev. Sci. Instruments* **89**, 101101 (2018).

748. Carter, C. B. & Williams, D. B. *Transmission Electron Microscopy: Diffraction, Imaging, and Spectrometry* (Springer, 2016).

749. Tang, C. & Yang, Z. Transmission Electron Microscopy (TEM). In *Membrane Characterization*, 145–159 (Elsevier, 2017).

750. Harris, J. R. Transmission Electron Microscopy in Molecular Structural Biology: A Historical Survey. *Arch. biochemistry biophysics* **581**, 3–18 (2015).

751. Herzog, C., Hook, D. & Konkiel, S. Dimensions: Bringing Down Barriers Between Scientometricians and Data. *Quant. Sci. Stud.* **1**, 387–395 (2020).

752. Bode, C., Herzog, C., Hook, D. & McGrath, R. A Guide to the Dimensions Data Approach. *Digit. Sci.* (2018).

753. Adams, J. *et al.* Dimensions-A Collaborative Approach to Enhancing Research Discovery. *Digit. Sci.* (2018).

754. Gleichmann, N. SEM vs TEM. Technology Networks: Analysis & Separations, Online: https://www.technologynetworks.com/analysis/articles/sem-vs-tem-331262 (2020).

755. Owen, G. Purchasing an Electron Microscope? – Considerations and Scientific Strategies to Help in the Decision Making Process. *Microscopy* (2018).

756. Electron Microscopy Suite: Price List. The Open University, Online: http://www9.open.ac.uk/emsuite/services/price-list (2020).

757. Electron Microscopy Research Services: Prices. Newcastle University, Online: https://www.ncl.ac.uk/emrs/prices (2020).

758. Sahlgrenska Academy: Prices for Electron Microscopy. University of Gothenburg, Online: https://cf.gu.se/english/centre_for_cellular_imaging/User_Information/Prices/electron-microscopy (2020).

759. Electron Microscopy: Pricelist. Harvard Medical School, Online: https://electron-microscopy.hms.harvard.edu/pricelist (2020).

760. Cambridge Advanced Imaging Centre: Services and Charges. University of Cambridge, Online: https://caic.bio.cam.ac.uk/booking/services (2020).

761. Ichimiya, A., Cohen, P. I. & Cohen, P. I. *Reflection High-Energy Electron Diffraction* (Cambridge University Press, 2004).

762. Braun, W. *Applied RHEED: Reflection High-Energy Electron Diffraction During Crystal Growth*, vol. 154 (Springer Science & Business Media, 1999).

763. Xiang, Y., Guo, F., Lu, T. & Wang, G. Reflection High-Energy Electron Diffraction Measurements of Reciprocal Space Structure of 2D Materials. *Nanotechnology* **27**, 485703 (2016).

764. Mašek, K., Moroz, V. & Matolín, V. Reflection High-Energy Electron Loss Spectroscopy (RHEELS): A New Approach in the Investigation of Epitaxial Thin Film Growth by Reflection High-Energy Electron Diffraction (RHEED). *Vacuum* **71**, 59–64 (2003).

765. Atwater, H. A. & Ahn, C. C. Reflection Electron Energy Loss Spectroscopy During Initial Stages of Ge Growth on Si by Molecular Beam Epitaxy. *Appl. Phys. Lett.* **58**, 269–271 (1991).

766. Yu, L. *et al.* Aberration Corrected Spin Polarized Low Energy Electron Microscope. *Ultramicroscopy* 113017 (2020).

767. Bauer, E. LEEM, SPLEEM and SPELEEM. In *Springer Handbook of Microscopy*, 2–2 (Springer, 2019).

768. Li, Q. *et al.* A Study of Chiral Magnetic Stripe Domains Within an In-Plane Virtual Magnetic Field Using SPLEEM. *APS* **2017**, L50–006 (2017).

769. Matsui, F. Auger Electron Spectroscopy. In *Compendium of Surface and Interface Analysis*, 39–44 (Springer, 2018).

770. MacDonald, N. & Waldrop, J. Auger Electron Spectroscopy in the Scanning Electron Microscope: Auger Electron Images. *Appl. Phys. Lett.* **19**, 315–318 (1971).

771. Scimeca, M., Bischetti, S., Lamsira, H. K., Bonfiglio, R. & Bonanno, E. Energy Dispersive X-Ray (EDX) Microanalysis: A Powerful Tool in Biomedical Research and Diagnosis. *Eur. J. Histochem.* **62** (2018).

772. Chen, Z. *et al.* Quantitative Atomic Resolution Elemental Mapping via Absolute-Scale Energy Dispersive X-Ray Spectroscopy. *Ultramicroscopy* **168**, 7–16 (2016).

773. Eggert, F., Camus, P., Schleifer, M. & Reinauer, F. Benefits from Bremsstrahlung Distribution Evaluation to get Unknown Information from Specimen in SEM and TEM. *IOP Conf. Series: Mater. Sci. Eng.* **304**, 012005 (2018).

774. Mohr, P. J., Newell, D. B. & Taylor, B. N. CODATA Recommended Values of the Fundamental Physical Constants: 2014. *J. Phys. Chem. Ref. Data* **45**, 043102 (2016).

775. Romano, A. & Marasco, A. An Introduction to Special Relativity. In *Classical Mechanics with Mathematica*®, 569–597 (Springer, 2018).

776. French, A. P. *Special Relativity* (CRC Press, 2017).

777. Rayleigh, L. XXXI. Investigations in Optics, with Special Reference to the Spectroscope. *The London, Edinburgh, Dublin Philos. Mag. J. Sci.* **8**, 261–274 (1879).

778. Ram, S., Ward, E. S. & Ober, R. J. Beyond Rayleigh's Criterion: A Resolution Measure with Application to Single-Molecule Microscopy. *Proc. Natl. Acad. Sci.* **103**, 4457–4462 (2006).

779. The Rayleigh Criterion. HyperPhysics, Online: http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/Raylei.html (2020).

780. Güémez, J., Fiolhais, M. & Fernández, L. A. The Principle of Relativity and the de Broglie Relation. *Am. J. Phys.* **84**, 443–447 (2016).

781. MacKinnon, E. De Broglie's Thesis: A Critical Retrospective. *Am. J. Phys.* **44**, 1047–1055 (1976).

782. DeBroglie Wavelength. HyperPhysics, Online: http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/debrog2.html#c5 (2020).

783. Glossary of TEM Terms: Wavelength of Electron. JEOL, Online: https://www.jeol.co.jp/en/words/emterms/search_result.html?keyword=wavelength%20of%20electron (2020).

784. Mendenhall, M. H. *et al.* High-Precision Measurement of the X-Ray Cu Kα Spectrum. *J. Phys. B: At. Mol. Opt. Phys.* **50**, 115004 (2017).

785. Transmission Electron Microscopy vs Scanning Electron Microscopy. ThermoFisher Scientific, Online: https://www.thermofisher.com/uk/en/home/materials-science/learning-center/applications/sem-tem-difference.html (2020).

786. Latychevskaia, T. Spatial Coherence of Electron Beams from Field Emitters and its Effect on the Resolution of Imaged Objects. *Ultramicroscopy* **175**, 121–129 (2017).

787. Van Dyck, D. Persistent Misconceptions about Incoherence in Electron Microscopy. *Ultramicroscopy* **111**, 894–900 (2011).

788. Krumeich, F. Properties of Electrons, their Interactions with Matter and Applications in Electron Microscopy. *Lab. Inorg. Chem.* 3–08 (2011).

789. Greffet, J.-J. & Nieto-Vesperinas, M. Field Theory for Generalized Bidirectional Reflectivity: Derivation of Helmholtz's Reciprocity Principle and Kirchhoff's Law. *JOSA A* **15**, 2735–2744 (1998).

790. Clarke, F. & Parry, D. Helmholtz Reciprocity: Its Validity and Application to Reflectometry. *Light. Res. & Technol.* **17**, 1–11 (1985).

791. Rose, H. & Kisielowski, C. F. On the Reciprocity of TEM and STEM. *Microsc. Microanal.* **11**, 2114 (2005).

792. Peters, J. J. P. *Structure and Ferroelectricity at the Atomic Level in Perovskite Oxides*. Ph.D. thesis, University of Warwick (2017).

793. Yakovlev, S., Downing, K., Wang, X. & Balsara, N. Advantages of HAADF vs. Conventional TEM Imaging for Study of PSS-PMB Diblock Copolymer Systems. *Microsc. Microanal.* **16**, 1698–1699 (2010).

794. Voelkl, E., Hoyle, D., Howe, J., Inada, H. & Yotsuji, T. STEM and TEM: Disparate Magnification Definitions and a Way Out. *Microsc. Microanal.* **23**, 56–57 (2017).

795. Bendersky, L. A. & Gayle, F. W. Electron Diffraction Using Transmission Electron Microscopy. *J. Res. Natl. Inst. Standards Technol.* **106**, 997 (2001).

796. Hubert, A., Römer, R. & Beanland, R. Structure Refinement from 'Digital' Large Angle Convergent Beam Electron Diffraction Patterns. *Ultramicroscopy* **198**, 1–9 (2019).

797. Beanland, R., Thomas, P. J., Woodward, D. I., Thomas, P. A. & Roemer, R. A. Digital Electron Diffraction – Seeing the Whole Picture. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **69**, 427–434 (2013).

798. Tanaka, M. Convergent-Beam Electron Diffraction. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **50**, 261–286 (1994).

799. Hovden, R. & Muller, D. A. Electron Tomography for Functional Nanomaterials. *arXiv preprint arXiv:2006.01652* (2020).

800. Koneti, S. *et al.* Fast Electron Tomography: Applications to Beam Sensitive Samples and in situ TEM or Operando Environmental TEM Studies. *Mater. Charact.* **151**, 480–495 (2019).

801. Song, H. *et al.* Electron Tomography: A Unique Tool Solving Intricate Hollow Nanostructures. *Adv. Mater.* **31**, 1801564 (2019).

802. Ercius, P., Alaidi, O., Rames, M. J. & Ren, G. Electron Tomography: A Three-Dimensional Analytic Tool for Hard and Soft Materials Research. *Adv. Mater.* **27**, 5638–5663 (2015).

803. Weyland, M. & Midgley, P. A. Electron Tomography. *Mater. Today* **7**, 32–40 (2004).

804. Wang, Z. *et al.* A Consensus Framework of Distributed Multiple-Tilt Reconstruction in Electron Tomography. *J. Comput. Biol.* **27**, 212–222 (2020).

805. Doerr, A. Cryo-Electron Tomography. *Nat. Methods* **14**, 34–34 (2017).

806. Öktem, O. Mathematics of Electron Tomography. *Handb. Math. Methods Imaging* **1** (2015).

807. Tichelaar, W., Hagen, W. J., Gorelik, T. E., Xue, L. & Mahamid, J. TEM Bright Field Imaging of Thick Specimens: Nodes in Thon Ring Patterns. *Ultramicroscopy* 113023 (2020).

808. Fujii, T. *et al.* Toward Quantitative Bright Field TEM Imaging of Ultra Thin Samples. *Microsc. Microanal.* **24**, 1612–1613 (2018).

809. Vander Wal, R. L. Soot Precursor Carbonization: Visualization Using LIF and LII and Comparison Using Bright and Dark Field TEM. *Combust. Flame* **112**, 607–616 (1998).

810. Bals, S., Kabius, B., Haider, M., Radmilovic, V. & Kisielowski, C. Annular Dark Field Imaging in a TEM. *Solid State Commun.* **130**, 675–680 (2004).

811. Yücelen, E., Lazić, I. & Bosch, E. G. Phase Contrast Scanning Transmission Electron Microscopy Imaging of Light and Heavy Atoms at the Limit of Contrast and Resolution. *Sci. Reports* **8**, 1–10 (2018).

812. Krajnak, M., McGrouther, D., Maneuski, D., O'Shea, V. & McVitie, S. Pixelated Detectors and Improved Efficiency for Magnetic Imaging in STEM Differential Phase Contrast. *Ultramicroscopy* **165**, 42–50 (2016).

813. Lazić, I., Bosch, E. G. & Lazar, S. Phase Contrast STEM for Thin Samples: Integrated Differential Phase Contrast. *Ultramicroscopy* **160**, 265–280 (2016).

814. Müller-Caspary, K. *et al.* Comparison of First Moment STEM with Conventional Differential Phase contrast and the Dependence on Electron Dose. *Ultramicroscopy* **203**, 95–104 (2019).

815. Zhou, D. *et al.* Sample Tilt Effects on Atom Column Position Determination in ABF-STEM Imaging. *Ultramicroscopy* **160**, 110–117 (2016).

816. Okunishi, E. *et al.* Visualization of Light Elements at Ultrahigh Resolution by STEM Annular Bright Field Microscopy. *Microsc. Microanal.* **15**, 164–165 (2009).

817. Van den Bos, K. H. *et al.* Unscrambling Mixed Elements Using High Angle Annular Dark Field Scanning Transmission Electron Microscopy. *Phys. Rev. Lett.* **116**, 246101 (2016).

818. McMullan, G., Faruqi, A. R. & Henderson, R. Direct Electron Detectors. In *Methods in Enzymology*, vol. 579, 1–17 (Elsevier, 2016).

819. McMullan, G., Chen, S., Henderson, R. & Faruqi, A. Detective Quantum Efficiency of Electron Area Detectors in Electron Microscopy. *Ultramicroscopy* **109**, 1126–1143 (2009).

820. Torruella, P. *et al.* Clustering Analysis Strategies for Electron Energy Loss Spectroscopy (EELS). *Ultramicroscopy* **185**, 42–48 (2018).

821. Pomarico, E. *et al.* Ultrafast Electron Energy-Loss Spectroscopy in Transmission Electron Microscopy. *Mrs Bull.* **43**, 497–503 (2018).

822. Koguchi, M., Tsuneta, R., Anan, Y. & Nakamae, K. Analytical Electron Microscope Based on Scanning Transmission Electron Microscope with Wavelength Dispersive X-Ray Spectroscopy to Realize Highly Sensitive Elemental Imaging Especially for Light Elements. *Meas. Sci. Technol.* **28**, 015904 (2016).

823. Tanaka, M., Takeguchi, M. & Furuya, K. X-Ray Analysis and Mapping by Wavelength Dispersive X-Ray Spectroscopy in an Electron Microscope. *Ultramicroscopy* **108**, 1427–1431 (2008).

824. Schwartz, A. J., Kumar, M., Adams, B. L. & Field, D. P. *Electron Backscatter Diffraction in Materials Science*, vol. 2 (Springer, 2009).

825. Humphreys, F. Review Grain and Subgrain Characterisation by Electron Backscatter Diffraction. *J. Mater. Sci.* **36**, 3833–3854 (2001).

826. Winkelmann, A., Nolze, G., Vos, M., Salvat-Pujol, F. & Werner, W. Physics-Based Simulation Models for EBSD: Advances and Challenges. *Nanoscale* **12**, 15 (2016).

827. Wright, S. I., Nowell, M. M. & Field, D. P. A Review of Strain Analysis Using Electron Backscatter Diffraction. *Microsc. Microanal.* **17**, 316 (2011).

828. Wilkinson, A. J., Meaden, G. & Dingley, D. J. High-Resolution Elastic Strain Measurement from Electron Backscatter Diffraction Patterns: New Levels of Sensitivity. *Ultramicroscopy* **106**, 307–313 (2006).

829. Kirkland, E. J. Image Simulation in Transmission Electron Microscopy. Cornell University, Online: http://muller.research.engineering.cornell.edu/sites/WEELS/summer06/mtutor.pdf (2006).

830. Kirkland, E. J. Computation in Electron Microscopy. *Acta Crystallogr. Sect. A: Foundations Adv.* **72**, 1–27 (2016).

831. Kirkland, E. J. *Advanced Computing in Electron Microscopy* (Springer Science & Business Media, 2010).

832. computem Repository. Online: https://sourceforge.net/projects/computem (2017).

833. Dyson, M. A. *Advances in Computational Methods for Transmission Electron Microscopy Simulation and Image Processing*. Ph.D. thesis, University of Warwick (2014).

834. Peters, J. J. P. & Dyson, M. A. clTEM. Online: https://github.com/JJPPeters/clTEM (2019).

835. cudaEM Repository. Online: https://github.com/ningustc/cudaEM (2018).

836. Barthel, J. Dr. Probe: A Software for High-Resolution STEM Image Simulation. *Ultramicroscopy* **193**, 1–11 (2018).

837. Barthel, J. Dr. Probe - STEM Image Simulation Software. Online: https://er-c.org/barthel/drprobe (2020).

838. Singh, S., Ram, F. & De Graef, M. EMsoft: Open Source Software for Electron Diffraction/Image Simulations. *Microsc. Microanal.* **23**, 212–213 (2017).

839. EMsoft Github Repository. Online: https://github.com/EMsoft-org/EMsoft (2020).

840. Stadelmann, P. JEMS. Online: https://web.archive.org/web/20151201081003/http://cimewww.epfl.ch/people/stadelmann/jemsWebSite/jems.html (2015).

841. Zuo, J. & Spence, J. *Electron Microdiffraction* (Springer Science & Business Media, 2013).

842. Lobato, I., Van Aert, S. & Verbeeck, J. Accurate and Fast Electron Microscopy Simulations Using the Open Source MULTEM Program. In *European Microscopy Congress 2016: Proceedings*, 531–532 (Wiley Online Library, 2016).

843. Lobato, I., Van Aert, S. & Verbeeck, J. Progress and New Advances in Simulating Electron Microscopy Datasets Using MULTEM. *Ultramicroscopy* **168**, 17–27 (2016).

844. Lobato, I. & Van Dyck, D. MULTEM: A New Multislice Program to Perform Accurate and Fast Electron Diffraction and Imaging Simulations Using Graphics Processing Units with CUDA. *Ultramicroscopy* **156**, 9–17 (2015).

845. O'Keefe, M. A. & Kilaas, R. Advances in High-Resolution Image Simulation. *Pfefferkorn Conf. Proceeding* (1988).

846. Electron Direct Methods. Online: http://www.numis.northwestern.edu/edm (2020).

847. Northwestern University Multislice and Imaging System. Online: http://www.numis.northwestern.edu/Software (2020).

848. Pryor, A., Ophus, C. & Miao, J. A Streaming Multi-GPU Implementation of Image Simulation Algorithms for Scanning Transmission Electron Eicroscopy. *Adv. Struct. Chem. Imaging* **3**, 15 (2017).

849. Ophus, C. A Fast Image Simulation Algorithm for Scanning Transmission Electron Microscopy. *Adv. Struct. Chem. Imaging* **3**, 13 (2017).

850. Prismatic Repository. Online: https://github.com/prism-em/prismatic (2020).

851. QSTEM. Online: https://www.physics.hu-berlin.de/en/sem/software/software_qstem (2020).

852. Gómez-Rodríguez, A., Beltrán-del Río, L. & Herrera-Becerra, R. SimulaTEM: Multislice Simulations for General Objects. *Ultramicroscopy* **110**, 95–104 (2010).

853. STEM-CELL. Online: http://tem-s3.nano.cnr.it/?page_id=2 (2020).

854. Tempas. Online: https://www.totalresolution.com/ (2020).

855. Ishizuka, K. A Practical Approach for STEM Image Simulation Based on the FFT Multislice Method. *Ultramicroscopy* **90**, 71–83 (2002).

856. Ishizuka, K. Prospects of Atomic Resolution Imaging with an Aberration-Corrected STEM. *Microscopy* **50**, 291–305 (2001).

857. Ishizuka, K. Multislice Formula for Inclined Illumination. *Acta Crystallogr. Sect. A: Cryst. Physics, Diffraction, Theor. Gen. Crystallogr.* **38**, 773–779 (1982).

858. Ishizuka, K. Contrast Transfer of Crystal Images in TEM. *Ultramicroscopy* **5**, 55–65 (1980).

859. Ishizuka, K. & Uyeda, N. A new theoretical and practical approach to the multislice method. *Acta Crystallogr. Sect. A: Cryst. Physics, Diffraction, Theor. Gen. Crystallogr.* **33**, 740–749 (1977).

860. HREM Simulation Suite. HREM Research, Online: https://www.hremresearch.com/Eng/simulation.html (2020).

861. Gianola, S., Jesus, T. S., Bargeri, S. & Castellini, G. Publish or Perish: Reporting Characteristics of Peer-Reviewed Publications, Pre-Prints and Registered Studies on the COVID-19 Pandemic. *medRxiv* (2020).

862. Nielsen, P. & Davison, R. M. Predatory Journals: A Sign of an Unhealthy Publish or Perish Game? *Inf. Syst. J.* (2020).

863. Génova, G. & de la Vara, J. L. The Problem is not Professional Publishing, but the Publish-or-Perish Culture. *Sci. Eng. Ethics* **25**, 617–619 (2019).

864. Zuo, J.-M. & Weickenmeier, A. On the Beam Selection and Convergence in the Bloch-Wave Method. *Ultramicroscopy* **57**, 375–383 (1995).

865. Yang, Y., Yang, Q., Huang, J., Cai, C. & Lin, J. Quantitative Comparison Between Real Space and Bloch Wave Methods in Image Simulation. *Micron* **100**, 73–78 (2017).

866. Peng, Y., Nellist, P. D. & Pennycook, S. J. HAADF-STEM Imaging with Sub-Angstrom Probes: A Full Bloch Wave Analysis. *J. Electron Microsc.* **53**, 257–266 (2004).

867. Cheng, L., Ming, Y. & Ding, Z. Bohmian Trajectory-Bloch Wave Approach to Dynamical Simulation of Electron Diffraction in Crystal. *New J. Phys.* **20**, 113004 (2018).

868. Beanland, R., Evans, K., Roemer, R. A. *et al.* Felix. Online: https://github.com/RudoRoemer/Felix (2020).

869. Morimura, T. & Hasaka, M. Bloch-Wave-Based STEM Image Simulation With Layer-by-Layer Representation. *Ultramicroscopy* **109**, 1203–1209 (2009).

870. Gatan Microscopy Suite Software. Online: www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software (2020).

871. FELMI/ZFE Script Database. Online: https://www.felmi-zfe.at/dm-script (2020).

872. Gatan Scripts Library. Online: https://www.gatan.com/resources/scripts-library (2020).

873. Potapov, P. temDM: Software for TEM in DigitalMicrograph. Online: http://temdm.com/web (2020).

874. Koch, C. Electron Microscopy Software. Online: https://www.physics.hu-berlin.de/en/sem/software (2016).

875. Schaffer, B. "How to script..." - Digital Micrograph Scripting Handbook. Online: http://digitalmicrograph-scripting.tavernmaker.de/HowToScript_index.htm (2015).

876. Mitchell, D. A Guide to Compiling C++ Code to Create Plugins for DigitalMicrograph (GMS 2.x). Dave Mitchell's DigitalMicrograph Scripting Website, Online: http://www.dmscripting.com/tutorial_compiling_plugins_for_GMS2.pdf (2014).

877. Miller, B. & Mick, S. Real-Time Data Processing using Python in DigitalMicrograph. *Microsc. Microanal.* **25**, 234–235 (2019).

878. Hoffman, C. RAM Disks Explained: What They Are and Why You Probably Shouldn't Use One. How-To Geek, Online: https://www.howtogeek.com/171432/ram-disks-explained-what-they-are-and-why-you-probably-shouldnt-use-one (2019).

879. Coughlin, T., Hoyt, R. & Handy, J. Digital Storage and Memory Technology (Part 1). IEEE Technology Trend Paper, https://www.ieee.org/content/dam/ieee-org/ieee/web/org/about/corporate/ieee-industry-advisory-board/digital-storage-memory-technology.pdf (2017).

880. A dedicated Site for Quantitative Electron Microscopy. HREM Research, Online: https://www.hremresearch.com/index.html (2020).

881. Rene de Cotret, L. P. TCP Socket Plug-In for Gatan Microscopy Suite 3.x. Online: https://github.com/LaurentRDC/gms-socket-plugin (2019).

882. Schorb, M., Haberbosch, I., Hagen, W. J., Schwab, Y. & Mastronarde, D. N. Software Tools for Automated Transmission Electron Microscopy. *Nat. Methods* **16**, 471–477 (2019).

883. Peters, J. J. P. DM Stack Builder. Online: https://github.com/JJPPeters/DM-Stack-Builder (2018).

884. Wolf, D., Lubk, A. & Lichte, H. Weighted Simultaneous Iterative Reconstruction Technique for Single-Axis Tomography. *Ultramicroscopy* **136**, 15–25 (2014).

885. Wolf, D. Tomography Menu. Online: http://wwwpub.zih.tu-dresden.de/~dwolf/ (2013).

886. Schindelin, J., Rueden, C. T., Hiner, M. C. & Eliceiri, K. W. The ImageJ Ecosystem: An Open Platform for Biomedical Image Analysis. *Mol. reproduction development* **82**, 518–529 (2015).

887. EM Software. EMDataResource, Online: https://www.emdataresource.org/emsoftware.html (2020).

888. Software Tools For Molecular Microscopy. WikiBooks, Online: https://en.wikibooks.org/wiki/Software_Tools_For_Molecular_Microscopy (2020).

889. Centre for Microscopy and Microanalysis: Online Tools: Scientific Freeware. University of Queensland, Online: https://cmm.centre.uq.edu.au/online-tools (2020).

890. Ben-Nun, T. & Hoefler, T. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. *ACM Comput. Surv. (CSUR)* **52**, 1–43 (2019).

891. Dryden, N. *et al.* Channel and Filter Parallelism for Large-Scale CNN Training. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–20 (2019).

892. Nwankpa, C., Ijomah, W., Gachagan, A. & Marshall, S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *arXiv preprint arXiv:1811.03378* (2018).

893. Hayou, S., Doucet, A. & Rousseau, J. On the Impact of the Activation Function on Deep Neural Networks Training. *arXiv preprint arXiv:1902.06853* (2019).

894. Roos, M. Deep Learning Neurons versus Biological Neurons. Towards Data Science, Online: https://towardsdatascience.com/deep-learning-versus-biological-neurons-floating-point-numbers-spikes-and-neurotransmitters-6eebfa3390e9 (2019).

895. Eldan, R. & Shamir, O. The Power of Depth for Feedforward Neural Networks. In *Conference on learning theory*, 907–940 (2016).

896. Telgarsky, M. Benefits of Depth in Neural Networks. *arXiv preprint arXiv:1602.04485* (2016).

897. Ba, J. & Caruana, R. Do Deep Nets Really Need to be Deep? In *Advances in neural information processing systems*, 2654–2662 (2014).

898. Lee, J. *et al.* Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In *Advances in Neural Information Processing Systems*, 8572–8583 (2019).

899. Yun, C., Sra, S. & Jadbabaie, A. Small Nonlinearities in Activation Functions Create Bad Local Minima in Neural Networks. *arXiv preprint arXiv:1802.03487* (2018).

900. Nair, V. & Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814 (2010).

901. Glorot, X., Bordes, A. & Bengio, Y. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323 (2011).

902. Maas, A. L., Hannun, A. Y. & Ng, A. Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proceedings of the International Conference on Machine Learning*, vol. 30, 3 (2013).

903. Chen, Y. *et al.* Dynamic ReLU. *arXiv preprint arXiv:2003.10027* (2020).

904. Xu, B., Wang, N., Chen, T. & Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv preprint arXiv:1505.00853* (2015).

905. Pedamonti, D. Comparison of Non-Linear Activation Functions for Deep Neural Networks on MNIST Classification Task. *arXiv preprint arXiv:1804.02763* (2018).

906. Chris. Leaky ReLU: improving traditional ReLU. MachineCurve, Online: https://www.machinecurve.com/index.php/2019/10/15/leaky-relu-improving-traditional-relu (2019).

907. Arnekvist, I., Carvalho, J. F., Kragic, D. & Stork, J. A. The Effect of Target Normalization and Momentum on Dying ReLU. *arXiv preprint arXiv:2005.06195* (2020).

908. Lu, L., Shin, Y., Su, Y. & Karniadakis, G. E. Dying ReLU and Initialization: Theory and Numerical Examples. *arXiv preprint arXiv:1903.06733* (2019).

909. Douglas, S. C. & Yu, J. Why RELU Units Sometimes Die: Analysis of Single-Unit Error Backpropagation in Neural Networks. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 864–868 (IEEE, 2018).

910. Krizhevsky, A. & Hinton, G. Convolutional Deep Belief Networks on CIFAR-10. *Tech. Rep.* **40**, 1–9 (2010).

911. Shang, W., Sohn, K., Almeida, D. & Lee, H. Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units. In *International Conference on Machine Learning*, 2217–2225 (2016).

912. Gao, H., Cai, L. & Ji, S. Adaptive Convolutional ReLUs. In *AAAI*, 3914–3921 (2020).

913. Eidnes, L. & Nøkland, A. Shifting Mean Activation Towards Zero with Bipolar Activation Functions. *arXiv preprint arXiv:1709.04054* (2017).

914. Jiang, X., Pang, Y., Li, X., Pan, J. & Xie, Y. Deep Neural Networks with Elastic Rectified Linear Units for Object Recognition. *Neurocomputing* **275**, 1132–1139 (2018).

915. Basirat, M. & ROTH, P. L* ReLU: Piece-wise Linear Activation Functions for Deep Fine-grained Visual Categorization. In *The IEEE Winter Conference on Applications of Computer Vision*, 1218–1227 (2020).

916. Clevert, D.-A., Unterthiner, T. & Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289* (2015).

917. Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. Self-Normalizing Neural Networks. In *Advances in Neural Information Processing Systems*, 971–980 (2017).

918. Hryniowski, A. & Wong, A. DeepLABNet: End-to-end Learning of Deep Radial Basis Networks with Fully Learnable Basis Functions. *arXiv preprint arXiv:1911.09257* (2019).

919. Dash, C. S. K., Behera, A. K., Dehuri, S. & Cho, S.-B. Radial Basis Function Neural Networks: A Topical State-of-the-Art Survey. *Open Comput. Sci.* **1** (2016).

920. Orr, M. J. L. Introduction to radial basis function networks. Online: https://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf (1996).

921. Jang, J.-S. & Sun, C.-T. Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems. *IEEE Transactions on Neural Networks* **4**, 156–159 (1993).

922. Wuraola, A. & Patel, N. Computationally Efficient Radial Basis Function. In *International Conference on Neural Information Processing*, 103–112 (Springer, 2018).

923. Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L. & Lopez, A. A Comprehensive Survey on Support Vector Machine Classification: Applications, Challenges and Trends. *Neurocomputing* (2020).

924. Scholkopf, B. & Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (Adaptive Computation and Machine Learning Series, 2018).

925. Tavara, S. Parallel Computing of Support Vector Machines: A Survey. *ACM Comput. Surv. (CSUR)* **51**, 1–38 (2019).

926. Kundu, A. *et al.* K-TanH: Hardware Efficient Activations For Deep Learning. *arXiv preprint arXiv:1909.07729* (2019).

927. LeCun, Y. A., Bottou, L., Orr, G. B. & Müller, K.-R. Efficient Backprop. In *Neural Networks: Tricks of the Trade*, 9–48 (Springer, 2012).

928. Abdelouahab, K., Pelcat, M. & Berry, F. Why TanH is a Hardware Friendly Activation Function for CNNs. In *Proceedings of the 11th International Conference on Distributed Smart Cameras*, 199–201 (2017).

929. Gulcehre, C., Moczulski, M., Denil, M. & Bengio, Y. Noisy Activation Functions. In *International Conference on Machine Learning*, 3059–3068 (2016).

930. Dunne, R. A. & Campbell, N. A. On the Pairing of the Softmax Activation and Cross-Entropy Penalty Functions and the Derivation of the Softmax Activation Function. In *Proceedings of the 8th Australian Conference on Neural Networks, Melbourne*, vol. 181, 185 (Citeseer, 1997).

931. Dumoulin, V. & Visin, F. A Guide to Convolution Arithmetic for Deep Learning. *arXiv preprint arXiv:1603.07285* (2018).

932. Graham, B. Fractional Max-Pooling. *arXiv preprint arXiv:1412.6071* (2014).

933. Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. Striving for Simplicity: The All Convolutional Net. *arXiv preprint arXiv:1412.6806* (2014).

934. Sabour, S., Frosst, N. & Hinton, G. E. Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems*, 3856–3866 (2017).

935. Luo, C. *et al.* Cosine Normalization: Using Cosine Similarity Instead of Dot Product in Neural Networks. In *International Conference on Artificial Neural Networks*, 382–391 (Springer, 2018).

936. Nader, A. & Azar, D. Searching for Activation Functions Using a Self-Adaptive Evolutionary Algorithm. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 145–146 (2020).

937. Ramachandran, P., Zoph, B. & Le, Q. Searching for Activation Functions. Google Research, Online: https://research.google/pubs/pub46503 (2018).

938. Bingham, G. & Miikkulainen, R. Discovering Parametric Activation Functions. *arXiv preprint arXiv:2006.03179* (2020).

939. Ertuğrul, Ö. F. A Novel Type of Activation Function in Artificial Neural Networks: Trained Activation Function. *Neural Networks* **99**, 148–157 (2018).

940. Lau, M. M. & Lim, K. H. Review of Adaptive Activation Function in Deep Neural Network. In *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 686–690 (IEEE, 2018).

941. Chung, H., Lee, S. J. & Park, J. G. Deep Neural Network Using Trainable Activation Functions. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 348–352 (IEEE, 2016).

942. Agostinelli, F., Hoffman, M., Sadowski, P. & Baldi, P. Learning Activation Functions to Improve Deep Neural Networks. *arXiv preprint arXiv:1412.6830* (2014).

943. Wu, Y., Zhao, M. & Ding, X. Beyond Weights Adaptation: A New Neuron Model with Trainable Activation Function and its Supervised Learning. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 2, 1152–1157 (IEEE, 1997).

944. Lee, J. *et al.* ProbAct: A Probabilistic Activation Function for Deep Neural Networks. *arXiv preprint arXiv:1905.10761* (2019).

945. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114* (2014).

946. Springenberg, J. T. & Riedmiller, M. Improving Deep Neural Networks with Probabilistic Maxout Units. *arXiv preprint arXiv:1312.6116* (2013).

947. Bawa, V. S. & Kumar, V. Linearized Sigmoidal Activation: A Novel Activation Function with Tractable Non-Linear Characteristics to Boost Representation Capability. *Expert. Syst. with Appl.* **120**, 346–356 (2019).

948. Kurita, K. An Overview of Normalization Methods in Deep Learning. Machine Learning Explained, Online: https://mlexplained.com/2018/11/30/an-overview-of-normalization-methods-in-deep-learning (2018).

949. Ren, M., Liao, R., Urtasun, R., Sinz, F. H. & Zemel, R. S. Normalizing the Normalizers: Comparing and Extending Network Normalization Schemes. *arXiv preprint arXiv:1611.04520* (2016).

950. Liao, Q., Kawaguchi, K. & Poggio, T. Streaming Normalization: Towards Simpler and More Biologically-Plausible Normalizations for Online and Recurrent Learning. *arXiv preprint arXiv:1610.06160* (2016).

951. Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A. How Does Batch Normalization Help Optimization? In *Advances in Neural Information Processing Systems*, 2483–2493 (2018).

952. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167* (2015).

953. Bjorck, N., Gomes, C. P., Selman, B. & Weinberger, K. Q. Understanding Batch Normalization. In *Advances in Neural Information Processing Systems*, 7694–7705 (2018).

954. Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J. & Schoenholz, S. S. A Mean Field Theory of Batch Normalization. *arXiv preprint arXiv:1902.08129* (2019).

955. Ioffe, S. & Cortes, C. Batch Normalization Layers (2019). US Patent 10,417,562.

956. Lian, X. & Liu, J. Revisit Batch Normalization: New Understanding and Refinement via Composition Optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 3254–3263 (2019).

957. Gao, P., Yu, L., Wu, Y. & Li, J. Low latency RNN Inference with Cellular Batching. In *Proceedings of the Thirteenth EuroSys Conference*, 1–15 (2018).

958. Fang, Z., Hong, D. & Gupta, R. K. Serving Deep Neural Networks at the Cloud Edge for Vision Applications on Mobile Platforms. In *Proceedings of the 10th ACM Multimedia Systems Conference*, 36–47 (2019).

959. Das, D. *et al.* Distributed Deep Learning using Synchronous Stochastic Gradient Descent. *arXiv preprint arXiv:1602.06709* (2016).

960. Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. & Tang, P. T. P. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv preprint arXiv:1609.04836* (2016).

961. Masters, D. & Luschi, C. Revisiting Small Batch Training for Deep Neural Networks. *arXiv preprint arXiv:1804.07612* (2018).

962. You, Y., Gitman, I. & Ginsburg, B. Scaling SGD Batch Size to 32k for ImageNet Training. Tech. Rep. UCB/EECS-2017-156, EECS Department, University of California, Berkeley (2017).

963. Devarakonda, A., Naumov, M. & Garland, M. AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks. *arXiv preprint arXiv:1712.02029* (2017).

964. Hoffer, E. *et al.* Augment Your Batch: Better Training With Larger Batches. *arXiv preprint arXiv:1901.09335* (2019).

965. Hasani, M. & Khotanlou, H. An Empirical Study on Position of the Batch Normalization Layer in Convolutional Neural Networks. In *2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, 1–4 (IEEE, 2019).

966. Mishkin, D. Batch Normalization Benchmarks. Online: https://github.com/ducha-aiki/caffenet-benchmark/blob/master/batchnorm.md (2016).

967. Nado, Z. *et al.* Evaluating Prediction-Time Batch Normalization for Robustness Under Covariate Shift. *arXiv preprint arXiv:2006.10963* (2020).

968. Zha, D., Lai, K.-H., Zhou, K. & Hu, X. Experience Replay Optimization. *arXiv preprint arXiv:1906.08387* (2019).

969. Schaul, T., Quan, J., Antonoglou, I. & Silver, D. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952* (2015).

970. Ioffe, S. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. In *Advances in Neural Information Processing Systems*, 1945–1953 (2017).

971. Salimans, T. *et al.* Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, 2234–2242 (2016).

972. Chiley, V. *et al.* Online Normalization for Training Neural Networks. In *Advances in Neural Information Processing Systems*, 8433–8443 (2019).

973. Hoffer, E., Banner, R., Golan, I. & Soudry, D. Norm Matters: Efficient and Accurate Normalization Schemes in Deep Networks. In *Advances in Neural Information Processing Systems*, 2160–2170 (2018).

974. Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer Normalization. *arXiv preprint arXiv:1607.06450* (2016).

975. Xu, J., Sun, X., Zhang, Z., Zhao, G. & Lin, J. Understanding and Improving Layer Normalization. In *Advances in Neural Information Processing Systems*, 4381–4391 (2019).

976. Ulyanov, D., Vedaldi, A. & Lempitsky, V. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv preprint arXiv:1607.08022* (2017).

977. Wu, Y. & He, K. Group Normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19 (2018).

978. Luo, P., Peng, Z., Ren, J. & Zhang, R. Do Normalization Layers in a Deep ConvNet Really Need to be Distinct? *arXiv preprint arXiv:1811.07727* (2018).

979. Luo, P., Ren, J., Peng, Z., Zhang, R. & Li, J. Differentiable Learning-to-Normalize Via Switchable Normalization. *arXiv preprint arXiv:1806.10779* (2018).

980. Nam, H. & Kim, H.-E. Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks. In *Advances in Neural Information Processing Systems*, 2558–2567 (2018).

981. Hao, K. We Analyzed 16,625 Papers to Figure Out Where AI is Headed Next. *MIT Technol. Rev.* (2019).

982. Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç. & Courville, A. Recurrent Batch Normalization. *arXiv preprint arXiv:1603.09025* (2016).

983. Liao, Q. & Poggio, T. Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex. *arXiv preprint arXiv:1604.03640* (2016).

984. Laurent, C., Pereyra, G., Brakel, P., Zhang, Y. & Bengio, Y. Batch Normalized Recurrent Neural Networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2657–2661 (IEEE, 2016).

985. Salimans, T. & Kingma, D. P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 901–909 (2016).

986. Qiao, S., Wang, H., Liu, C., Shen, W. & Yuille, A. Weight Standardization. *arXiv preprint arXiv:1903.10520* (2019).

987. Gitman, I. & Ginsburg, B. Comparison of Batch Normalization and Weight Normalization Algorithms for the Large-Scale Image Classification. *arXiv preprint arXiv:1709.08145* (2017).

988. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. *arXiv preprint arXiv:1802.05957* (2018).

989. Wood, G. R. & Zhang, B. P. Estimation of the Lipschitz Constant of a Function. *J. Glob. Optim.* **8**, 91–103 (1996).

990. Gui, J., Sun, Z., Wen, Y., Tao, D. & Ye, J. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *arXiv preprint arXiv:2001.06937* (2020).

991. Saxena, D. & Cao, J. Generative Adversarial Networks (gans): Challenges, Solutions, and Future Directions. *arXiv preprint arXiv:2005.00065* (2020).

992. Pan, Z. *et al.* Recent Progress on Generative Adversarial Networks (GANs): A Survey. *IEEE Access* **7**, 36322–36333 (2019).

993. Wang, Z., She, Q. & Ward, T. E. Generative Adversarial Networks: A Survey and Taxonomy. *arXiv preprint arXiv:1906.01529* (2019).

994. Hui, J. Machine Learning — Singular Value Decomposition (SVD) & Principal Component Analysis (PCA). Medium, Online: https://medium.com/@jonathan_hui/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491 (2019).

995. Afham, M. Singular Value Decomposition and its Applications in Principal Component Analysis. Towards Data Science, Online: https://towardsdatascience.com/singular-value-decomposition-and-its-applications-in-principal-component-analysis-5b7a5f08d0bd (2020).

996. Wall, M. E., Rechtsteiner, A. & Rocha, L. M. Singular Value Decomposition and Principal Component Analysis. In *A Practical Approach to Microarray Data Analysis*, 91–109 (Springer, 2003).

997. Klema, V. & Laub, A. The Singular Value Decomposition: Its Computation and Some Applications. *IEEE Transactions on Autom. Control.* **25**, 164–176 (1980).

998. Yoshida, Y. & Miyato, T. Spectral Norm Regularization for Improving the Generalizability of Deep Learning. *arXiv preprint arXiv:1705.10941* (2017).

999. Golub, G. H. & Van der Vorst, H. A. Eigenvalue Computation in the 20th Century. *J. Comput. Appl. Math.* **123**, 35–65 (2000).

1000. Nguyen, T. Q. & Salazar, J. Transformers Without Tears: Improving the Normalization of Self-Attention. *arXiv preprint arXiv:1910.05895* (2019).

1001. Nguyen, T. Q. & Chiang, D. Improving Lexical Choice in Neural Machine Translation. *arXiv preprint arXiv:1710.01329* (2017).

1002. Stewart, M. Simple Introduction to Convolutional Neural Networks. Towards Data Science, Online: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac (2019).

1003. Wu, J. Introduction to Convolutional Neural Networks. *Natl. Key Lab for Nov. Softw. Technol.* **5**, 23 (2017).

1004. McCann, M. T., Jin, K. H. & Unser, M. Convolutional Neural Networks for Inverse Problems in Imaging: A Review. *IEEE Signal Process. Mag.* **34**, 85–95 (2017).

1005. O'Shea, K. & Nash, R. An Introduction to Convolutional Neural Networks. *arXiv preprint arXiv:1511.08458* (2015).

1006. Hubel, D. H. & Wiesel, T. N. Receptive Fields and Functional Architecture of Monkey Striate Cortex. *The J. Physiol.* **195**, 215–243 (1968).

1007. Fukushima, K. A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biol. Cybern.* **36**, 193–202 (1980).

1008. Fukushima, K. & Miyake, S. Neocognitron: A Self-Organizing Neural Network Nodel for a Mechanism of Visual Pattern Recognition. In *Competition and Cooperation in Neural Nets*, 267–285 (Springer, 1982).

1009. Fukushima, K. Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition. *Neural Networks* **1**, 119–130 (1988).

1010. Fukushima, K. Neocognitron for Handwritten Digit Recognition. *Neurocomputing* **51**, 161–180 (2003).

1011. Atlas, L. E., Homma, T. & Marks II, R. J. An Artificial Neural Network for Spatio-Temporal Bipolar Patterns: Application to Phoneme Classification. In *Neural Information Processing Systems*, 31–40 (1988).

1012. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **86**, 2278–2324 (1998).

1013. LeCun, Y., Haffner, P., Bottou, L. & Bengio, Y. Object Recognition with Gradient-Based Learning. In *Shape, Contour and Grouping in Computer Vision*, 319–345 (Springer, 1999).

1014. Cireşan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Comput.* **22**, 3207–3220 (2010).

1015. Yao, G., Lei, T. & Zhong, J. A Review of Convolutional-Neural-Network-Based Action Recognition. *Pattern Recognit. Lett.* **118**, 14–22 (2019).

1016. Gupta, A. *et al.* Deep Learning in Image Cytometry: A Review. *Cytom. Part A* **95**, 366–380 (2019).

1017. Ma, S. *et al.* Image and Video Compression with Neural Networks: A Review. *IEEE Transactions on Circuits Syst. for Video Technol.* (2019).

1018. Liu, D., Li, Y., Lin, J., Li, H. & Wu, F. Deep Learning-Based Video Coding: A Review and a Case Study. *ACM Comput. Surv. (CSUR)* **53**, 1–35 (2020).

1019. Bouwmans, T., Javed, S., Sultana, M. & Jung, S. K. Deep Neural Network Concepts for Background Subtraction: A Systematic Review and Comparative Evaluation. *Neural Networks* (2019).

1020. Jing, Y. *et al.* Neural Style Transfer: A Review. *IEEE Transactions on Vis. Comput. Graph.* (2019).

1021. Anwar, S. M. *et al.* Medical Image Analysis using Convolutional Neural Networks: A Review. *J. Med. Syst.* **42**, 226 (2018).

1022. Soffer, S. *et al.* Convolutional Neural Networks for Radiologic Images: A Radiologist's Guide. *Radiology* **290**, 590–606 (2019).

1023. Yamashita, R., Nishio, M., Do, R. K. G. & Togashi, K. Convolutional Neural Networks: An Overview and Application in Radiology. *Insights into Imaging* **9**, 611–629 (2018).

1024. Bernal, J. *et al.* Deep Convolutional Neural Networks for Brain Image Analysis on Magnetic Resonance Imaging: A Review. *Artif. Intell. Medicine* **95**, 64–81 (2019).

1025. Fu, Y. *et al.* Deep Learning in Medical Image Registration: A Review. *Phys. Medicine & Biol.* (2020).

1026. Badar, M., Haris, M. & Fatima, A. Application of Deep Learning for Retinal Image Analysis: A review. *Comput. Sci. Rev.* **35**, 100203 (2020).

1027. Litjens, G. *et al.* A Survey on Deep Learning in Medical Image Analysis. *Med. Image Analysis* **42**, 60–88 (2017).

1028. Liu, J. *et al.* Applications of Deep Learning to MRI Images: A Survey. *Big Data Min. Anal.* **1**, 1–18 (2018).

1029. Zhao, Z.-Q., Zheng, P., Xu, S.-t. & Wu, X. Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks Learn. Syst.* **30**, 3212–3232 (2019).

1030. Wang, W. *et al.* Salient Object Detection in the Deep Learning Era: An In-Depth Survey. *arXiv preprint arXiv:1904.09146* (2019).

1031. Minaee, S. *et al.* Deep Learning Based Text Classification: A Comprehensive Review. *arXiv preprint arXiv:2004.03705* (2020).

1032. TensorFlow Core v2.2.0 Python Documentation for Convolutional Layer. Online: https://web.archive.org/web/20200520184050/https://www.tensorflow.org/api_docs/python/tf/nn/convolution (2020).

1033. McAndrew, A. *A Computational Introduction to Digital Image Processing* (CRC Press, 2015).

1034. Smoothing Images. OpenCV Documentation, Online: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html (2019).

1035. Vairalkar, M. K. & Nimbhorkar, S. Edge Detection of Images Using Sobel Operator. *Int. J. Emerg. Technol. Adv. Eng.* **2**, 291–293 (2012).

1036. Bogdan, V., Bonchiş, C. & Orhei, C. Custom Extended Sobel Filters. *arXiv preprint arXiv:1910.00138* (2019).

1037. Jähne, B., Scharr, H., Körkel, S. *et al.* Principles of filter design. *Handb. Comput. Vis. Appl.* **2**, 125–151 (1999).

1038. Scharr, H. *Optimal Operators in Digital Image Processing (in German)*. Ph.D. thesis, University of Heidelberg (2000).

1039. Kawalec-Latała, E. Edge Detection on Images of Pseudoimpedance Section Supported by Context and Adaptive Transformation Model Images. *Studia Geotech. et Mech.* **36**, 29–36 (2014).

1040. Roberts, L. G. *Machine Perception of Three-Dimensional Solids*. Ph.D. thesis, Massachusetts Institute of Technology (1963).

1041. Prewitt, J. M. Object Enhancement and Extraction. *Pict. Process. Psychopictorics* **10**, 15–19 (1970).

1042. Jin, J., Dundar, A. & Culurciello, E. Flattened Convolutional Neural Networks for Feedforward Acceleration. *arXiv preprint arXiv:1412.5474* (2014).

1043. Chen, J., Lu, Z., Xue, J.-H. & Liao, Q. XSepConv: Extremely Separated Convolution. *arXiv preprint arXiv:2002.12046* (2020).

1044. Jaderberg, M., Vedaldi, A. & Zisserman, A. Speeding up Convolutional Neural Networks with Low Rank Expansions. *arXiv preprint arXiv:1405.3866* (2014).

1045. Wu, S., Wang, G., Tang, P., Chen, F. & Shi, L. Convolution With Even-Sized Kernels and Symmetric Padding. In *Advances in Neural Information Processing Systems*, 1194–1205 (2019).

1046. Kossaifi, J., Bulat, A., Panagakis, Y., Pantic, M. & Cambridge, S. A. Efficient *N*-Dimensional Convolutions via Higher-Order Factorization. *arXiv preprint arXiv:1906.06196* (2019).

1047. Chris. Using Constant Padding, Reflection Padding and Replication Padding with Keras. MachineCurve, Online: https://www.machinecurve.com/index.php/2020/02/10/Using-constant-padding-reflection-padding-and-replication-padding-with-keras (2020).

1048. Liu, G. *et al.* Partial Convolution Based Padding. *arXiv preprint arXiv:1811.11718* (2018).

1049. Larsson, G., Maire, M. & Shakhnarovich, G. FractalNet: Ultra-Deep Neural Networks Without Residuals. *arXiv preprint arXiv:1605.07648* (2016).

1050. Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

1051. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826 (2016).

1052. Szegedy, C. *et al.* Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9 (2015).

1053. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. Learning Transferable Architectures for Scalable Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710 (2018).

1054. Kim, J., Kwon Lee, J. & Mu Lee, K. Deeply-Recursive Convolutional Network for Image Super-Resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1637–1645 (2016).

1055. Tai, Y., Yang, J. & Liu, X. Image Super-Resolution via Deep Recursive Residual Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3147–3155 (2017).

1056. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778 (2016).

1057. Dwarampudi, M. & Reddy, N. Effects of Padding on LSTMs and CNNs. *arXiv preprint arXiv:1903.07288* (2019).

1058. Liu, G. *et al.* Image Inpainting for Irregular Holes Using Partial Convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 85–100 (2018).

1059. Peng, Z. Multilayer Perceptron Algebra. *arXiv preprint arXiv:1701.04968* (2017).

1060. Pratama, M., Za'in, C., Ashfahani, A., Ong, Y. S. & Ding, W. Automatic Construction of Multi-Layer Perceptron Network from Streaming Examples. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1171–1180 (2019).

1061. Neyshabur, B. Towards Learning Convolutions from Scratch. *arXiv preprint arXiv:2007.13657* (2020).

1062. Guo, L., Liu, F., Cai, C., Liu, J. & Zhang, G. 3D Deep Encoder-Decoder Network for Fluorescence Molecular Tomography. *Opt. Lett.* **44**, 1892–1895 (2019).

1063. Oseledets, I. V. Tensor-Train Decomposition. *SIAM J. on Sci. Comput.* **33**, 2295–2317 (2011).

1064. Novikov, A., Podoprikhin, D., Osokin, A. & Vetrov, D. P. Tensorizing Neural Networks. In *Advances in Neural Information Processing Systems*, 442–450 (2015).

1065. Kong, C. & Lucey, S. Take it in Your Stride: Do We Need Striding in CNNs? *arXiv preprint arXiv:1712.02502* (2017).

1066. Zaniolo, L. & Marques, O. On The Use of Variable Stride in Convolutional Neural Networks. *Multimed. Tools Appl.* 1–18 (2020).

1067. Shi, W. *et al.* Is the Deconvolution Layer the Same as a Convolutional Layer? *arXiv preprint arXiv:1609.07009* (2016).

1068. Aitken, A. *et al.* Checkerboard Artifact Free Sub-Pixel Convolution: A Note on Sub-Pixel Convolution, Resize Convolution and Convolution Resize. *arXiv preprint arXiv:1707.02937* (2017).

1069. Odena, A., Dumoulin, V. & Olah, C. Deconvolution and Checkerboard Artifacts. *Distill* **1** (2016).

1070. Howard, A. G. *et al.* MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861* (2017).

1071. Guo, J., Li, Y., Lin, W., Chen, Y. & Li, J. Network Decoupling: From Regular to Depthwise Separable Convolutions. *arXiv preprint arXiv:1808.05517* (2018).

1072. Depthwise Separable Convolutional Neural Networks. GeeksforGeeks, Online: https://www.geeksforgeeks.org/depth-wise-separable-convolutional-neural-networks (2020).

1073. Liu, T. Depth-wise Separable Convolutions: Performance Investigations. Online: https://tlkh.dev/depsep-convs-perf-investigations (2020).

1074. Gunther, L. The Eye. In *The Physics of Music and Color*, 325–335 (Springer, 2019).

1075. Lamb, T. D. Why Rods and Cones? *Eye* **30**, 179–185 (2016).

1076. Cohen, A. I. Rods and Cones. In *Physiology of Photoreceptor Organs*, 63–110 (Springer, 1972).

1077. He, K., Zhang, X., Ren, S. & Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis Mach. Intell.* **37**, 1904–1916 (2015).

1078. Zhang, D.-Q. Image Recognition Using Scale Recurrent Neural Networks. *arXiv preprint arXiv:1803.09218* (2018).

1079. Tanaka, N. Introduction to Fourier Transforms for TEM and STEM. In *Electron Nano-Imaging*, 219–226 (Springer, 2017).

1080. Mathematica. Fourier Transform Conventions. Online: https://reference.wolfram.com/language/tutorial/Calculus.html#26017 (2020).

1081. Frigo, M. & Johnson, S. G. The Design and Implementation of FFTW3. *Proc. IEEE* **93**, 216–231 (2005).

1082. Stokfiszewski, K., Wieloch, K. & Yatsymirskyy, M. The Fast Fourier Transform Partitioning Scheme for GPU's Computation Effectiveness Improvement. In *Conference on Computer Science and Information Technologies*, 511–522 (Springer, 2017).

1083. Chen, Y., Cui, X. & Mei, H. Large-Scale FFT on GPU Clusters. In *Proceedings of the 24th ACM International Conference on Supercomputing*, 315–324 (2010).

1084. Gu, L., Li, X. & Siegel, J. An Empirically Tuned 2D and 3D FFT Library on CUDA GPU. In *Proceedings of the 24th ACM International Conference on Supercomputing*, 305–314 (2010).

1085. Puchała, D., Stokfiszewski, K., Yatsymirskyy, M. & Szczepaniak, B. Effectiveness of Fast Fourier Transform Implementations on GPU and CPU. In *2015 16th International Conference on Computational Problems of Electrical Engineering (CPEE)*, 162–164 (IEEE, 2015).

1086. Ogata, Y., Endo, T., Maruyama, N. & Matsuoka, S. An Efficient, Model-Based CPU-GPU Heterogeneous FFT Library. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, 1–10 (IEEE, 2008).

1087. Cooley, J. W. & Tukey, J. W. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. computation* **19**, 297–301 (1965).

1088. Duhamel, P. & Vetterli, M. Fast Fourier Transforms: A Tutorial Review and A State of the Art. *Signal Process. (Elsevier)* **19**, 259–299 (1990).

1089. clFFT Repository. Online: https://github.com/clMathLibraries/clFFT (2017).

1090. Highlander, T. & Rodriguez, A. Very Efficient Training of Convolutional Neural Networks Using Fast Fourier Transform and Overlap-and-Add. *arXiv preprint arXiv:1601.06815* (2016).

1091. Weisstein, E. W. Convolution Theorem. Wolfram Mathworld – A Wolfram Web Resource, Online: https://mathworld.wolfram.com/ConvolutionTheorem.html (2020).

1092. Pratt, H., Williams, B., Coenen, F. & Zheng, Y. FCNN: Fourier Convolutional Neural Networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 786–798 (Springer, 2017).

1093. Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).

1094. Thomson, A. M. Neocortical Layer 6, A Review. *Front. Neuroanat.* **4**, 13 (2010).

1095. Fitzpatrick, D. The Functional Organization of Local Circuits in Visual Cortex: Insights From the Study of Tree Shrew Striate Cortex. *Cereb. Cortex* **6**, 329–341 (1996).

1096. Zaeemzadeh, A., Rahnavard, N. & Shah, M. Norm-Preservation: Why Residual Networks can Become Extremely Deep? *IEEE Transactions on Pattern Analysis Mach. Intell.* (2020).

1097. Kawaguchi, K. & Bengio, Y. Depth with Nonlinearity Creates No Bad Local Minima in ResNets. *Neural Networks* **118**, 167–174 (2019).

1098. Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Visualizing the Loss Landscape of Neural Nets. In *Advances in Neural Information Processing Systems*, 6389–6399 (2018).

1099. Veit, A., Wilber, M. J. & Belongie, S. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. In *Advances in Neural Information Processing Systems*, 550–558 (2016).

1100. Greff, K., Srivastava, R. K. & Schmidhuber, J. Highway and Residual Networks Learn Unrolled Iterative Estimation. *arXiv preprint arXiv:1612.07771* (2016).

1101. Martinez, J., Hossain, R., Romero, J. & Little, J. J. A Simple Yet Effective Baseline for 3D Human Pose Estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2640–2649 (2017).

1102. Yue, B., Fu, J. & Liang, J. Residual Recurrent Neural Networks for Learning Sequential Representations. *Information* **9**, 56 (2018).

1103. Kim, J., El-Khamy, M. & Lee, J. Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition. In *Proceedings of Interspeech 2017*, 1591–1595 (2017).

1104. Wu, Y. *et al.* Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* (2016).

1105. Srivastava, R. K., Greff, K. & Schmidhuber, J. Training Very Deep Networks. In *Advances in Neural Information Processing Systems*, 2377–2385 (2015).

1106. Srivastava, R. K., Greff, K. & Schmidhuber, J. Highway Networks. *arXiv preprint arXiv:1505.00387* (2015).

1107. Huang, G., Liu, Z., Pleiss, G., Van Der Maaten, L. & Weinberger, K. Convolutional Networks with Dense Connectivity. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2019).

1108. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708 (2017).

1109. Tong, T., Li, G., Liu, X. & Gao, Q. Image Super-Resolution Using Dense Skip Connections. In *Proceedings of the IEEE International Conference on Computer Vision*, 4799–4807 (2017).

1110. Jiang, F. *et al.* An End-to-End Compression Framework Based on Convolutional Neural Networks. *IEEE Transactions on Circuits Syst. for Video Technol.* **28**, 3007–3018 (2017).

1111. Yang, G. & Schoenholz, S. Mean Field Residual Networks: On the Edge of Chaos. In *Advances in Neural Information Processing Systems*, 7103–7114 (2017).

1112. Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S. & Pennington, J. Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks. In *International Conference on Machine Learning*, 5393–5402 (2018).

1113. Wu, Q. & Wang, F. Concatenate Convolutional Neural Networks for Non-Intrusive Load Monitoring Across Complex Background. *Energies* **12**, 1572 (2019).

1114. Terwilliger, A. M., Perdue, G. N., Isele, D., Patton, R. M. & Young, S. R. Vertex Reconstruction of Neutrino Interactions Using Deep Learning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 2275–2281 (IEEE, 2017).

1115. Gers, F. A., Schraudolph, N. N. & Schmidhuber, J. Learning Precise Timing with LSTM Recurrent Networks. *J. Mach. Learn. Res.* **3**, 115–143 (2002).

1116. Gers, F. A. & Schmidhuber, E. LSTM Recurrent Networks Learn Simple Context-Free and Context-Sensitive Languages. *IEEE Transactions on Neural Networks* **12**, 1333–1340 (2001).

1117. Lin, M., Chen, Q. & Yan, S. Network-in-Network. *arXiv preprint arXiv:1312.4400* (2013).

1118. Vaswani, A. *et al.* Attention is All You Need. In *Advances in Neural Information Processing Systems*, 5998–6008 (2017).

1119. Alammar, J. The Illustrated Transformer. GitHub Blog, Online: http://jalammar.github.io/illustrated-transformer (2018).

1120. Mnih, V., Heess, N., Graves, A. & Kavukcuoglu, K. Recurrent Models of Visual Attention. In *Advances in Neural Information Processing Systems*, 2204–2212 (2014).

1121. Ba, J., Mnih, V. & Kavukcuoglu, K. Multiple Object Recognition with Visual Attention. *arXiv preprint arXiv:1412.7755* (2014).

1122. Lillicrap, T. P. *et al.* Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971* (2015).

1123. Heess, N., Hunt, J. J., Lillicrap, T. P. & Silver, D. Memory-Based Control with Recurrent Neural Networks. *arXiv preprint arXiv:1512.04455* (2015).

1124. Konda, V. R. & Tsitsiklis, J. N. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, 1008–1014 (2000).

1125. Grabocka, J., Scholz, R. & Schmidt-Thieme, L. Learning Surrogate Losses. *arXiv preprint arXiv:1905.10108* (2019).

1126. Neftci, E. O., Mostafa, H. & Zenke, F. Surrogate Gradient Learning in Spiking Neural Networks. *IEEE Signal Process. Mag.* **36**, 61–63 (2019).

1127. Liang, K. J., Li, C., Wang, G. & Carin, L. Generative Adversarial Network Training is a Continual Learning Problem. *arXiv preprint arXiv:1811.11083* (2018).

1128. Jaderberg, M. *et al.* Decoupled Neural Interfaces Using Synthetic Gradients. In *International Conference on Machine Learning*, 1627–1635 (2017).

1129. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE transactions on image processing* **13**, 600–612 (2004).

1130. Pan, Z. *et al.* Loss Functions of Generative Adversarial Networks (GANs): Opportunities and Challenges. *IEEE Transactions on Emerg. Top. Comput. Intell.* (2020).

1131. Dong, H.-W. & Yang, Y.-H. Towards a Deeper Understanding of Adversarial Losses. *arXiv preprint arXiv:1901.08753* (2019).

1132. Mescheder, L., Geiger, A. & Nowozin, S. Which Training Methods for GANs do Actually Converge? *arXiv preprint arXiv:1801.04406* (2018).

1133. Kurach, K., Lučić, M., Zhai, X., Michalski, M. & Gelly, S. A Large-Scale Study on Regularization and Normalization in GANs. In *International Conference on Machine Learning*, 3581–3590 (2019).

1134. Roth, K., Lucchi, A., Nowozin, S. & Hofmann, T. Stabilizing Training of Generative Adversarial Networks Through Regularization. In *Advances in Neural Information Processing Systems*, 2018–2028 (2017).

1135. Goodfellow, I. *et al.* Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, 2672–2680 (2014).

1136. Mao, X. *et al.* On the Effectiveness of Least Squares Generative Adversarial Networks. *IEEE Transactions on Pattern Analysis Mach. Intell.* **41**, 2947–2960 (2018).

1137. Mao, X. *et al.* Least Squares Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2794–2802 (2017).

1138. Wiatrak, M. & Albrecht, S. V. Stabilizing Generative Adversarial Network Training: A Survey. *arXiv preprint arXiv:1910.00927* (2019).

1139. Bang, D. & Shim, H. MGGAN: Solving Mode Collapse Using Manifold Guided Training. *arXiv preprint arXiv:1804.04391* (2018).

1140. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, 214–223 (2017).

1141. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, 5767–5777 (2017).

1142. Hazan, T., Papandreou, G. & Tarlow, D. *Adversarial Perturbations of Deep Neural Networks*, 311–342 (MIT Press, 2017).

1143. Chen, Z., Badrinarayanan, V., Lee, C.-Y. & Rabinovich, A. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. *arXiv preprint arXiv:1711.02257* (2017).

1144. Lee, S. & Son, Y. Multitask Learning with Single Gradient Step Update for Task Balancing. *arXiv preprint arXiv:2005.09910* (2020).

1145. Zhang, H., Goodfellow, I., Metaxas, D. & Odena, A. Self-Attention Generative Adversarial Networks. In *International Conference on Machine Learning*, 7354–7363 (2019).

1146. Brock, A., Donahue, J. & Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv preprint arXiv:1809.11096* (2018).

1147. Hindupur, A. The GAN Zoo. Online: https://github.com/hindupuravinash/the-gan-zoo (2018).

1148. Wang, T.-C. *et al.* High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8798–8807 (2018).

1149. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232 (2017).

1150. Bashkirova, D., Usman, B. & Saenko, K. Unsupervised Video-to-Video Translation. *arXiv preprint arXiv:1806.03698* (2018).

1151. Liu, M.-Y., Breuel, T. & Kautz, J. Unsupervised Image-to-Image Translation Networks. In *Advances in Neural Information Processing Systems*, 700–708 (2017).

1152. Amodio, M. & Krishnaswamy, S. TraVeLGAN: Image-to-Image Translation by Transformation Vector Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8983–8992 (2019).

1153. Tzeng, E., Hoffman, J., Saenko, K. & Darrell, T. Adversarial Discriminative Domain Adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7167–7176 (2017).

1154. Ganin, Y. & Lempitsky, V. Unsupervised Domain Adaptation by Backpropagation. In *International Conference on Machine Learning*, 1180–1189 (2015).

1155. Tzeng, E., Hoffman, J., Darrell, T. & Saenko, K. Simultaneous Deep Transfer Across Domains and Tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, 4068–4076 (2015).

1156. Werbos, P. J. Backpropagation Through Time: What It Does and How To Do It. *Proc. IEEE* **78**, 1550–1560 (1990).

1157. Saldi, N., Yüksel, S. & Linder, T. Asymptotic Optimality of Finite Model Approximations for Partially Observed Markov Decision Processes With Discounted Cost. *IEEE Transactions on Autom. Control.* **65**, 130–142 (2019).

1158. Jaakkola, T., Singh, S. P. & Jordan, M. I. Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. In *Advances in Neural Information Processing Systems*, 345–352 (1995).

1159. Xu, K. *et al.* Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning*, 2048–2057 (2015).

1160. Vinyals, O., Toshev, A., Bengio, S. & Erhan, D. Show and Tell: A Neural Image Caption Generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3156–3164 (2015).

1161. Basmatkar, P., Holani, H. & Kaushal, S. Survey on Neural Machine Translation for Multilingual Translation System. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 443–448 (IEEE, 2019).

1162. Wu, S. *et al.* Deep Learning in Clinical Natural Language Processing: A Methodical Review. *J. Am. Med. Informatics Assoc.* **27**, 457–470 (2020).

1163. Otter, D. W., Medina, J. R. & Kalita, J. K. A Survey of the Usages of Deep Learning for Natural Language Processing. *IEEE Transactions on Neural Networks Learn. Syst.* (2020).

1164. Iyer, S. R., An, U. & Subramanian, L. Forecasting Sparse Traffic Congestion Patterns Using Message-Passing RNNs. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3772–3776 (IEEE, 2020).

1165. Mandal, P. K. & Mahto, R. Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection. In *16th International Conference on Information Technology-New Generations (ITNG 2019)*, 495–498 (Springer, 2019).

1166. Rhanoui, M., Mikram, M., Yousfi, S. & Barzali, S. A CNN-BiLSTM Model for Document-Level Sentiment Analysis. *Mach. Learn. Knowl. Extr.* **1**, 832–847 (2019).

1167. Zhang, X., Chen, F. & Huang, R. A Combination of RNN and CNN for Attention-Based Relation Classification. *Procedia Comput. Sci.* **131**, 911–917 (2018).

1168. Qu, Y., Liu, J., Kang, L., Shi, Q. & Ye, D. Question Answering Over Freebase via Attentive RNN with Similarity Matrix Based CNN. *arXiv preprint arXiv:1804.03317* **38** (2018).

1169. Sieg, A. From Pre-trained Word Embeddings To Pre-trained Language Models – Focus on BERT. Towards Data Science, Online: https://towardsdatascience.com/from-pre-trained-word-embeddings-to-pre-trained-language-models-focus-on-bert-343815627598 (2019).

1170. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).

1171. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, 3111–3119 (2013).

1172. Mnih, A. & Kavukcuoglu, K. Learning Word Embeddings Efficiently with Noise-Contrastive Estimation. In *Advances in Neural Information Processing Systems*, 2265–2273 (2013).

1173. Grave, É., Bojanowski, P., Gupta, P., Joulin, A. & Mikolov, T. Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (2018).

1174. Le, Q. & Mikolov, T. Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning*, 1188–1196 (2014).

1175. Lau, J. H. & Baldwin, T. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *arXiv preprint arXiv:1607.05368* (2016).

1176. Pennington, J., Socher, R. & Manning, C. D. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543 (2014).

1177. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

1178. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *Phys. D: Nonlinear Phenom.* **404**, 132306 (2020).

1179. Olah, C. Understanding LSTM Networks. Online: https://colah.github.io/posts/2015-08-Understanding-LSTMs (2015).

1180. Gers, F. A., Schmidhuber, J. & Cummins, F. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* **12**, 2451–2471 (2000).

1181. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **9**, 1735–1780 (1997).

1182. Cho, K. *et al.* Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078* (2014).

1183. Dey, R. & Salemt, F. M. Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1597–1600 (IEEE, 2017).

1184. Heck, J. C. & Salem, F. M. Simplified Minimal Gated Unit Variations for Recurrent Neural Networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1593–1596 (IEEE, 2017).

1185. Pascanu, R., Mikolov, T. & Bengio, Y. On the Difficulty of Training Recurrent Neural Networks. In *International Conference on Machine Learning*, 1310–1318 (2013).

1186. Hanin, B. Which Neural Net Architectures Give Rise to Exploding and Vanishing Gradients? In *Advances in Neural Information Processing Systems*, 582–591 (2018).

1187. Britz, D., Goldie, A., Luong, M.-T. & Le, Q. Massive Exploration of Neural Machine Translation Architectures. *arXiv preprint arXiv:1703.03906* (2017).

1188. Jozefowicz, R., Zaremba, W. & Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In *International Conference on Machine Learning*, 2342–2350 (2015).

1189. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS 2014 Workshop on Deep Learning* (2014).

1190. Gruber, N. & Jockisch, A. Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text? *Front. Artif. Intell.* **3**, 40 (2020).

1191. Weiss, G., Goldberg, Y. & Yahav, E. On the Practical Computational Power of Finite Precision RNNs for Language Recognition. *arXiv preprint arXiv:1805.04908* (2018).

1192. Bayer, J., Wierstra, D., Togelius, J. & Schmidhuber, J. Evolving Memory Cell Structures for Sequence Learning. In *International Conference on Artificial Neural Networks*, 755–764 (Springer, 2009).

1193. Zhou, G.-B., Wu, J., Zhang, C.-L. & Zhou, Z.-H. Minimal Gated Unit for Recurrent Neural Networks. *Int. J. Autom. Comput.* **13**, 226–234 (2016).

1194. Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks Learn. Syst.* **28**, 2222–2232 (2016).

1195. Mozer, M. C., Kazakov, D. & Lindsey, R. V. Discrete Event, Continuous Time RNNs. *arXiv preprint arXiv:1710.04110* (2017).

1196. Funahashi, K.-i. & Nakamura, Y. Approximation of Dynamical Systems by Continuous Time Recurrent Neural Networks. *Neural Networks* **6**, 801–806 (1993).

1197. Quinn, M. Evolving Communication Without Dedicated Communication Channels. In *European Conference on Artificial Life*, 357–366 (Springer, 2001).

1198. Beer, R. D. The Dynamics of Adaptive Behavior: A Research Program. *Robotics Auton. Syst.* **20**, 257–289 (1997).

1199. Harvey, I., Husbands, P. & Cliff, D. Seeing the Light: Artificial Evolution, Real Vision. *From Animals to Animat.* **3**, 392–401 (1994).

1200. Elman, J. L. Finding Structure in Time. *Cogn. Sci.* **14**, 179–211 (1990).

1201. Jordan, M. I. Serial Order: A Parallel Distributed Processing Approach. In *Advances in Psychology*, vol. 121, 471–495 (Elsevier, 1997).

1202. Li, S., Li, W., Cook, C., Zhu, C. & Gao, Y. Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5457–5466 (2018).

1203. Sathasivam, S. & Abdullah, W. A. T. W. Logic Learning in Hopfield Networks. *arXiv preprint arXiv:0804.4075* (2008).

1204. Tutschku, K. Recurrent Multilayer Perceptrons for Identification and Control: The Road to Applications. *Inst. Comput. Sci. Res. Report, Univ. Würzburg Am Hubland* (1995).

1205. Jia, Y., Wu, Z., Xu, Y., Ke, D. & Su, K. Long Short-Term Memory Projection Recurrent Neural Network Architectures for Piano's Continuous Note Recognition. *J. Robotics* **2017** (2017).

1206. Pascanu, R., Gulcehre, C., Cho, K. & Bengio, Y. How to Construct Deep Recurrent Neural Networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)* (2014).

1207. Schuster, M. & Paliwal, K. K. Bidirectional Recurrent Neural Networks. *IEEE transactions on Signal Process.* **45**, 2673–2681 (1997).

1208. Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015* (2015).

1209. Graves, A. & Schmidhuber, J. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks* **18**, 602–610 (2005).

1210. Thireou, T. & Reczko, M. Bidirectional Long Short-Term Memory Networks for Predicting the Subcellular Localization of Eukaryotic Proteins. *IEEE/ACM Transactions on Comput. Biol. Bioinforma.* **4**, 441–446 (2007).

1211. Cho, K., Van Merriënboer, B., Bahdanau, D. & Bengio, Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv preprint arXiv:1409.1259* (2014).

1212. Zhang, T., Huang, M. & Zhao, L. Learning Structured Representation for Text Classification via Reinforcement Learning. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

1213. Chung, J., Ahn, S. & Bengio, Y. Hierarchical Multiscale Recurrent Neural Networks. *arXiv preprint arXiv:1609.01704* (2016).

1214. Sordoni, A. *et al.* A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 553–562 (2015).

1215. Paine, R. W. & Tani, J. How Hierarchical Control Self-Organizes in Artificial Adaptive Systems. *Adapt. Behav.* **13**, 211–225 (2005).

1216. Schmidhuber, J. Learning Complex, Extended Sequences Using the Principle of History Compression. *Neural Comput.* **4**, 234–242 (1992).

1217. Yamashita, Y. & Tani, J. Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: A Humanoid Robot Experiment. *PLoS Comput. Biol.* **4**, e1000220 (2008).

1218. Shibata Alnajjar, F., Yamashita, Y. & Tani, J. The Hierarchical and Functional Connectivity of Higher-Order Cognitive Mechanisms: Neurorobotic Model to Investigate the Stability and Flexibility of Working Memory. *Front. Neurorobotics* **7**, 2 (2013).

1219. Chaudhari, S., Polatkan, G., Ramanath, R. & Mithal, V. An Attentive Survey of Attention Models. *arXiv preprint arXiv:1904.02874* (2019).

1220. Luong, M.-T., Pham, H. & Manning, C. D. Effective Approaches to Attention-Based Neural Machine Translation. *arXiv preprint arXiv:1508.04025* (2015).

1221. Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473* (2014).

1222. Graves, A. *et al.* Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature* **538**, 471–476 (2016).

1223. Graves, A., Wayne, G. & Danihelka, I. Neural Turing Machines. *arXiv preprint arXiv:1410.5401* (2014).

1224. Tschannen, M., Bachem, O. & Lucic, M. Recent Advances in Autoencoder-Based Representation Learning. *arXiv preprint arXiv:1812.05069* (2018).

1225. Hinton, G. E. & Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *science* **313**, 504–507 (2006).

1226. Kramer, M. A. Nonlinear Principal Component Analysis Using Autoassociative Neural Networks. *AIChE J.* **37**, 233–243 (1991).

1227. Zhou, Y., Arpit, D., Nwogu, I. & Govindaraju, V. Is Joint Training Better for Deep Auto-Encoders? *arXiv preprint arXiv:1405.1380* (2014).

1228. Jolliffe, I. T. & Cadima, J. Principal Component Analysis: A Review and Recent Developments. *Philos. Transactions Royal Soc. A: Math. Phys. Eng. Sci.* **374**, 20150202 (2016).

1229. Theis, L., Shi, W., Cunningham, A. & Huszár, F. Lossy Image Compression with Compressive Autoencoders. *arXiv preprint arXiv:1703.00395* (2017).

1230. Vincent, P. *et al.* Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* **11** (2010).

1231. Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103 (2008).

1232. Gondara, L. Medical Image Denoising Using Convolutional Denoising Autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 241–246 (IEEE, 2016).

1233. Cho, K. Simple Sparsification Improves Sparse Denoising Autoencoders in Denoising Highly Corrupted Images. In *International Conference on Machine Learning*, 432–440 (2013).

1234. Cho, K. Boltzmann Machines and Denoising Autoencoders for Image Denoising. *arXiv preprint arXiv:1301.3468* (2013).

1235. Rifai, S., Vincent, P., Muller, X., Glorot, X. & Bengio, Y. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *International Conference on Machine Learning* (2011).

1236. Rifai, S. *et al.* Higher Order Contractive Auto-Encoder. In *Joint European conference on Machine Learning and Knowledge Discovery in Databases*, 645–660 (Springer, 2011).

1237. Kingma, D. P. & Welling, M. An Introduction to Variational Autoencoders. *arXiv preprint arXiv:1906.02691* (2019).

1238. Doersch, C. Tutorial on Variational Autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

1239. Makhzani, A. & Frey, B. *k*-Sparse Autoencoders. *arXiv preprint arXiv:1312.5663* (2013).

1240. Nair, V. & Hinton, G. E. 3D Object Recognition with Deep Belief Nets. In *Advances in Neural Information Processing Systems*, 1339–1347 (2009).

1241. Arpit, D., Zhou, Y., Ngo, H. & Govindaraju, V. Why Regularized Auto-Encoders Learn Sparse Representation? In *International Conference on Machine Learning*, 136–144 (2016).

1242. Zeng, N. *et al.* Facial Expression Recognition via Learning Deep Sparse Autoencoders. *Neurocomputing* **273**, 643–649 (2018).

1243. Yin, Y., Ouyang, L., Wu, Z. & Yin, S. A Survey of Generative Adversarial Networks Based on Encoder-Decoder Model. *Math. Comput. Sci.* **5**, 31 (2020).

1244. Yu, X., Zhang, X., Cao, Y. & Xia, M. VAEGAN: A Collaborative Filtering Framework Based on Adversarial Variational Autoencoders. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 4206–4212 (AAAI Press, 2019).

1245. Larsen, A. B. L., Sønderby, S. K., Larochelle, H. & Winther, O. Autoencoding Beyond Pixels Using a Learned Similarity Metric. In *International Conference on Machine Learning*, 1558–1566 (2016).

1246. Zhuang, F. & Moulin, P. A New Variational Method for Deep Supervised Semantic Image Hashing. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4532–4536 (IEEE, 2020).

1247. Jin, G., Zhang, Y. & Lu, K. Deep Hashing Based on VAE-GAN for Efficient Similarity Retrieval. *Chin. J. Electron.* **28**, 1191–1197 (2019).

1248. Khobahi, S. & Soltanalian, M. Model-Aware Deep Architectures for One-Bit Compressive Variational Autoencoding. *arXiv preprint arXiv:1911.12410* (2019).

1249. Wang, B., Liu, K. & Zhao, J. Deep Semantic Hashing with Multi-Adversarial Training. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1453–1462 (2018).

1250. Patterson, N. & Wang, Y. Semantic Hashing with Variational Autoencoders.

1251. Fan, Y. *et al.* Video Anomaly Detection and Localization via Gaussian Mixture Fully Convolutional Variational Autoencoder. *Comput. Vis. Image Underst.* 102920 (2020).

1252. Yao, R., Liu, C., Zhang, L. & Peng, P. Unsupervised Anomaly Detection Using Variational Auto-Encoder based Feature Extraction. In *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 1–7 (IEEE, 2019).

1253. Xu, H. *et al.* Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In *Proceedings of the 2018 World Wide Web Conference*, 187–196 (2018).

1254. An, J. & Cho, S. Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability. *Special Lect. on IE* **2**, 1–18 (2015).

1255. Gauerhof, L. & Gu, N. Reverse Variational Autoencoder for Visual Attribute Manipulation and Anomaly Detection. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2103–2112 (IEEE, 2020).

1256. Klys, J., Snell, J. & Zemel, R. Learning Latent Subspaces in Variational Autoencoders. In *Advances in Neural Information Processing Systems*, 6444–6454 (2018).

1257. Borysov, S. S., Rich, J. & Pereira, F. C. How to Generate Micro-Agents? A Deep Generative Modeling Approach to Population Synthesis. *Transp. Res. Part C: Emerg. Technol.* **106**, 73–97 (2019).

1258. Salim Jr, A. Synthetic Patient Generation: A Deep Learning Approach Using Variational Autoencoders. *arXiv preprint arXiv:1808.06444* (2018).

1259. Gómez-Bombarelli, R. *et al.* Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **4**, 268–276 (2018).

1260. Zhavoronkov, A. *et al.* Deep Learning Enables Rapid Identification of Potent DDR1 Kinase Inhibitors. *Nat. Biotechnol.* **37**, 1038–1040 (2019).

1261. Griffiths, R.-R. & Hernández-Lobato, J. M. Constrained Bayesian Optimization for Automatic Chemical Design Using Variational Autoencoders. *Chem. Sci.* **11**, 577–586 (2020).

1262. Lim, J., Ryu, S., Kim, J. W. & Kim, W. Y. Molecular Generative Model Based on Conditional Variational Autoencoder for *de novo* Molecular Design. *J. Cheminformatics* **10**, 1–9 (2018).

1263. Wan, Z., Zhang, Y. & He, H. Variational Autoencoder Based Synthetic Data Generation for Imbalanced Learning. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–7 (IEEE, 2017).

1264. Zhang, J. M., Harman, M., Ma, L. & Liu, Y. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Softw. Eng.* (2020).

1265. Amershi, S. *et al.* Software Engineering for Machine Learning: A Case Study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 291–300 (IEEE, 2019).

1266. Breck, E., Cai, S., Nielsen, E., Salib, M. & Sculley, D. The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction. In *2017 IEEE International Conference on Big Data (Big Data)*, 1123–1132 (IEEE, 2017).

1267. Sculley, D. *et al.* Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*, 2503–2511 (2015).

1268. Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Loss landscape mit license. Online: https://github.com/tomgoldstein/loss-landscape/blob/master/LICENSE (2017).

1269. Rodríguez, O. H. & Lopez Fernandez, J. M. A Semiotic Reflection on the Didactics of the Chain Rule. *The Math. Enthus.* **7**, 321–332 (2010).

1270. Kiefer, J. & Wolfowitz, J. Stochastic Estimation of the Maximum of a Regression Function. *The Annals Math. Stat.* **23**, 462–466 (1952).

1271. Robbins, H. & Monro, S. A Stochastic Approximation Method. *The Annals Math. Stat.* 400–407 (1951).

1272. Polyak, B. T. Some Methods of Speeding up the Convergence of Iteration Methods. *USSR Comput. Math. Math. Phys.* **4**, 1–17 (1964).

1273. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the Importance of Initialization and Momentum in Deep Learning. In *International Conference on Machine Learning*, 1139–1147 (2013).

1274. Su, W., Boyd, S. & Candes, E. A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights. In *Advances in Neural Information Processing Systems*, 2510–2518 (2014).

1275. TensorFlow Source Code for Nesterov Momentum. Online: https://github.com/tensorflow/tensorflow/blob/23c218785eac5bfe737eec4f8081fd0ef8e0684d/tensorflow/python/training/momentum_test.py#L40 (2018).

1276. Ma, J. & Yarats, D. Quasi-Qyperbolic Momentum and ADAM for Deep Learning. *arXiv preprint arXiv:1810.06801* (2018).

1277. Lucas, J., Sun, S., Zemel, R. & Grosse, R. Aggregated Momentum: Stability Through Passive Damping. *arXiv preprint arXiv:1804.00325* (2018).

1278. Hinton, G., Srivastava, N. & Swersky, K. Neural Networks for Machine Learning Lecture 6a Overview of Mini-Batch Gradient Descent. Online: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (2012).

1279. Kingma, D. P. & Ba, J. ADAM: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

1280. Sun, S., Cao, Z., Zhu, H. & Zhao, J. A Survey of Optimization Methods from a Machine Learning Perspective. *IEEE Transactions on Cybern.* (2019).

1281. Bottou, L., Curtis, F. E. & Nocedal, J. Optimization Methods for Large-Scale Machine Learning. *Siam Rev.* **60**, 223–311 (2018).

1282. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747* (2016).

1283. Curry, H. B. The Method of Steepest Descent for Non-Linear Minimization Problems. *Q. Appl. Math.* **2**, 258–261 (1944).

1284. Lemaréchal, C. Cauchy and the Gradient Method. *Documenta Math. Extra* **251**, 254 (2012).

1285. Chen, T., Xu, B., Zhang, C. & Guestrin, C. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174* (2016).

1286. Cybertron AI. Saving Memory Using Gradient-Checkpointing. Online: https://github.com/cybertronai/gradient-checkpointing (2019).

1287. Jin, P., Ginsburg, B. & Keutzer, K. Spatially Parallel Convolutions. *OpenReview.net* (2018).

1288. Whittington, J. C. R. & Bogacz, R. Theories of Error Back-Propagation in the Brain. *Trends Cogn. Sci.* **23**, 235–250 (2019).

1289. Green, C. S. & Bavelier, D. Exercising Your Brain: A Review of Human Brain Plasticity and Training-Induced Learning. *Psychol. Aging* **23**, 692 (2008).

1290. Bassett, D. S. *et al.* Dynamic Reconfiguration of Human Brain Networks During Learning. *Proc. Natl. Acad. Sci.* **108**, 7641–7646 (2011).

1291. O'Doherty, J. P. Reward Representations and Reward-Related Learning in the Human Brain: Insights from Neuroimaging. *Curr. Opin. Neurobiol.* **14**, 769–776 (2004).

1292. Luo, L., Xiong, Y., Liu, Y. & Sun, X. Adaptive Gradient Methods with Dynamic Bound of Learning Rate. *arXiv preprint arXiv:1902.09843* (2019).

1293. Reddi, S. J., Kale, S. & Kumar, S. On the Convergence of ADAM and Beyond. *arXiv preprint arXiv:1904.09237* (2019).

1294. Zhang, M., Lucas, J., Ba, J. & Hinton, G. E. Lookahead Optimizer: *k* Steps Forward, 1 Step Back. In *Advances in Neural Information Processing Systems*, 9597–9608 (2019).

1295. Dozat, T. Incorporating Nesterov Momentum into ADAM. OpenReview, Online: https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ (2016).

1296. Huang, H., Wang, C. & Dong, B. Nostalgic Adam: Weighting More of the Past Gradients When Designing the Adaptive Learning Rate. *arXiv preprint arXiv:1805.07557* (2018).

1297. Baiesi, M. Power Gradient Descent. *arXiv preprint arXiv:1906.04787* (2019).

1298. Liu, L. *et al.* On the Variance of the Adaptive Learning Rate and Beyond. *arXiv preprint arXiv:1908.03265* (2019).

1299. Bello, I., Zoph, B., Vasudevan, V. & Le, Q. V. Neural Optimizer Search with Reinforcement Learning. *arXiv preprint arXiv:1709.07417* (2017).

1300. Andrychowicz, M. *et al.* Learning to Learn by Gradient Descent by Gradient Descent. In *Advances in Neural Information Processing Systems*, 3981–3989 (2016).

1301. Li, K. & Malik, J. Learning to Optimize. *arXiv preprint arXiv:1606.01885* (2016).

1302. Hochreiter, S., Younger, A. S. & Conwell, P. R. Learning to Learn Using Gradient Descent. In *International Conference on Artificial Neural Networks*, 87–94 (Springer, 2001).

1303. Duan, Y. *et al.* RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv preprint arXiv:1611.02779* (2016).

1304. Zou, D., Cao, Y., Zhou, D. & Gu, Q. Stochastic Gradient Descent Optimizes Over-Parameterized Deep relu Networks. *arXiv preprint arXiv:1811.08888* (2018).

1305. Watt, J. Two Natural Weaknesses of Gradient Descent. Online: https://jermwatt.github.io/machine_learning_refined/notes/3_First_order_methods/3_7_Problems.html (2020).

1306. Goh, G. Why Momentum Really Works. *Distill* (2017).

1307. Qian, N. On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Networks* **12**, 145–151 (1999).

1308. Schmidt, R. M., Schneider, F. & Hennig, P. Descending Through a Crowded Valley – Benchmarking Deep Learning Optimizers. *arXiv preprint arXiv:2007.01547* (2020).

1309. Choi, D. *et al.* On Empirical Comparisons of Optimizers for Deep Learning. *arXiv preprint arXiv:1910.05446* (2019).

1310. Wilson, A. C., Roelofs, R., Stern, M., Srebro, N. & Recht, B. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In *Advances in Neural Information Processing Systems*, 4148–4158 (2017).

1311. Dogo, E., Afolabi, O., Nwulu, N., Twala, B. & Aigbavboa, C. A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks. In *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, 92–99 (IEEE, 2018).

1312. Seetharaman, P., Wichern, G., Pardo, B. & Roux, J. L. AutoClip: Adaptive Gradient Clipping for Source Separation Networks. *arXiv preprint arXiv:2007.14469* (2020).

1313. Gorbunov, E., Danilova, M. & Gasnikov, A. Stochastic Optimization with Heavy-Tailed Noise via Accelerated Gradient Clipping. *arXiv preprint arXiv:2005.10785* (2020).

1314. Yoshida, T. & Ohki, K. Natural Images are Reliably Represented by Sparse and Variable Populations of Neurons in Visual Cortex. *Nat. Commun.* **11**, 1–19 (2020).

1315. Probst, P., Bischl, B. & Boulesteix, A.-L. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *arXiv preprint arXiv:1802.09596* (2018).

1316. Ge, R., Kakade, S. M., Kidambi, R. & Netrapalli, P. The Step Decay Schedule: A Near Optimal, Geometrically Decaying Learning Rate Procedure. *arXiv preprint arXiv:1904.12838* (2019).

1317. Chen, J. & Kyrillidis, A. Decaying Momentum Helps Neural Network Training. *arXiv preprint arXiv:1910.04952* (2019).

1318. Yang, L. & Shami, A. On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. *arXiv preprint arXiv:2007.15745* (2020).

1319. Chandra, K. *et al.* Gradient Descent: The Ultimate Optimizer. *arXiv preprint arXiv:1909.13371* (2019).

1320. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631 (2019).

1321. Lakhmiri, D., Digabel, S. L. & Tribes, C. HyperNOMAD: Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search. *arXiv preprint arXiv:1907.01698* (2019).

1322. Ilievski, I., Akhtar, T., Feng, J. & Shoemaker, C. A. Efficient Hyperparameter Optimization of Deep Learning Algorithms Using Deterministic RBF Surrogates. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 822–829 (AAAI Press, 2017).

1323. Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S. & Pastor, J. R. Particle Swarm Optimization for Hyper-Parameter Selection in Deep Neural Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 481–488 (2017).

1324. Wilamowski, B. M. & Yu, H. Neural Network Learning Without Backpropagation. *IEEE Transactions on Neural Networks* **21**, 1793–1803 (2010).

1325. Blum, A., Dan, C. & Seddighin, S. Learning Complexity of Simulated Annealing. *arXiv preprint arXiv:2003.02981* (2020).

1326. Ingber, L. Simulated Annealing: Practice versus Theory. *Math. Comput. Model.* **18**, 29–57 (1993).

1327. Ayumi, V., Rere, L. R., Fanany, M. I. & Arymurthy, A. M. Optimization of Convolutional Neural Network Using Microcanonical Annealing Algorithm. In *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 506–511 (IEEE, 2016).

1328. Rere, L. M. R., Fanany, M. I. & Arymurthy, A. M. Simulated Annealing Algorithm for Deep Learning. *Procedia Comput. Sci.* **72**, 137–144 (2015).

1329. Borysenko, O. & Byshkin, M. CoolMomentum: A Method for Stochastic Optimization by Langevin Dynamics with Simulated Annealing. *arXiv preprint arXiv:2005.14605* (2020).

1330. Fischetti, M. & Stringher, M. Embedded Hyper-Parameter Tuning by Simulated Annealing. *arXiv preprint arXiv:1906.01504* (2019).

1331. Sloss, A. N. & Gustafson, S. 2019 Evolutionary Algorithms Review. In *Genetic Programming Theory and Practice XVII*, 307–344 (Springer, 2020).

1332. Al-Sahaf, H. *et al.* A Survey on Evolutionary Machine Learning. *J. Royal Soc. New Zealand* **49**, 205–228 (2019).

1333. Shapiro, J. Genetic Algorithms in Machine Learning. In *Advanced Course on Artificial Intelligence*, 146–168 (Springer, 1999).

1334. Doerr, B., Le, H. P., Makhmara, R. & Nguyen, T. D. Fast genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 777–784 (2017).

1335. Such, F. P. *et al.* Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv preprint arXiv:1712.06567* (2017).

1336. Sehgal, A., La, H., Louis, S. & Nguyen, H. Deep Reinforcement Learning using Genetic Algorithm for Parameter Optimization. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 596–601 (IEEE, 2019).

1337. Hu, C., Zuo, Y., Chen, C., Ong, S. P. & Luo, J. Genetic Algorithm-Guided Deep Learning of Grain Boundary Diagrams: Addressing the Challenge of Five Degrees of Freedom. *Mater. Today* (2020).

1338. Jennings, P. C., Lysgaard, S., Hummelshøj, J. S., Vegge, T. & Bligaard, T. Genetic Algorithms for Computational Materials Discovery Accelerated by Machine Learning. *npj Comput. Mater.* **5**, 1–6 (2019).

1339. Nigam, A., Friederich, P., Krenn, M. & Aspuru-Guzik, A. Augmenting Genetic Algorithms with Deep Neural Networks for Exploring the Chemical Space. *arXiv preprint arXiv:1909.11655* (2019).

1340. Potapov, A. & Rodionov, S. Genetic Algorithms with DNN-Based Trainable Crossover as an Example of Partial Specialization of General Search. In *International Conference on Artificial General Intelligence*, 101–111 (Springer, 2017).

1341. Powell, M. J. Direct Search Algorithms for Optimization Calculations. *Acta numerica* 287–336 (1998).

1342. Ranganathan, V. & Natarajan, S. A New Backpropagation Algorithm Without Gradient Descent. *arXiv preprint arXiv:1802.00027* (2018).

1343. Junior, F. E. F. & Yen, G. G. Particle Swarm Optimization of Deep Neural Networks Architectures for Image Classification. *Swarm Evol. Comput.* **49**, 62–74 (2019).

1344. Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A. & Benhaddou, D. Parameters Optimization of Deep Learning Models Using Particle Swarm Optimization. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 1285–1290 (IEEE, 2017).

1345. Kennedy, J. & Eberhart, R. Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1942–1948 (IEEE, 1995).

1346. Kennedy, J. The Particle Swarm: Social Adaptation of Knowledge. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, 303–308 (IEEE, 1997).

1347. Xu, Y. A Review of Machine Learning With Echo State Networks. *Proj. Rep.* (2020).

1348. Jaeger, H. Echo State Network. *Scholarpedia* **2**, 2330 (2007).

1349. Gallicchio, C. & Micheli, A. Deep Echo State Network (DeepESN): A Brief Survey. *arXiv preprint arXiv:1712.04323* (2017).

1350. Alaba, P. A. *et al.* Towards a More Efficient and Cost-Sensitive Extreme Learning Machine: A State-of-the-Art Review of Recent Trend. *Neurocomputing* **350**, 70–90 (2019).

1351. Ghosh, S. *et al.* A Survey on Extreme Learning Machine and Evolution of Its Variants. In *International Conference on Recent Trends in Image Processing and Pattern Recognition*, 572–583 (Springer, 2018).

1352. Albadra, M. A. A. & Tiuna, S. Extreme Learning Machine: A Review. *Int. J. Appl. Eng. Res.* **12**, 4610–4623 (2017).

1353. Tang, J., Deng, C. & Huang, G.-B. Extreme Learning Machine for Multilayer Perceptron. *IEEE Transactions on Neural Networks Learn. Syst.* **27**, 809–821 (2015).

1354. Huang, G.-B., Zhou, H., Ding, X. & Zhang, R. Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Transactions on Syst. Man, Cybern. Part B (Cybernetics)* **42**, 513–529 (2011).

1355. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme Learning Machine: Theory and Applications. *Neurocomputing* **70**, 489–501 (2006).

1356. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 2, 985–990 (IEEE, 2004).

1357. Li, Y. Deep Reinforcement Learning: An Overview. *arXiv preprint arXiv:1701.07274* (2017).

1358. Mondal, A. K. & Jamali, N. A Survey of Reinforcement Learning Techniques: Strategies, Recent Development, and Future Directions. *arXiv preprint arXiv:2001.06921* (2020).

1359. Haney, B. S. Deep Reinforcement Learning Patents: An Empirical Survey. *Available at SSRN 3570254* (2020).

1360. Nguyen, T. T., Nguyen, N. D. & Nahavandi, S. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Transactions on Cybern.* (2020).

1361. Botvinick, M. *et al.* Reinforcement Learning, Fast and Slow. *Trends Cogn. Sci.* **23**, 408–422 (2019).

1362. Recht, B. A Tour of Reinforcement Learning: The View From Continuous Control. *Annu. Rev. Control. Robotics, Auton. Syst.* **2**, 253–279 (2019).

1363. Arulkumaran, K., Deisenroth, M. P., Brundage, M. & Bharath, A. A. A Brief Survey of Deep Reinforcement Learning. *arXiv preprint arXiv:1708.05866* (2017).

1364. Kiran, B. R. *et al.* Deep Reinforcement Learning for Autonomous Driving: A Survey. *arXiv preprint arXiv:2002.00444* (2020).

1365. Nageshrao, S., Tseng, H. E. & Filev, D. Autonomous Highway Driving Using Deep Reinforcement Learning. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2326–2331 (IEEE, 2019).

1366. Talpaert, V. *et al.* Exploring Applications of Deep Reinforcement Learning for Real-World Autonomous Driving Systems. *arXiv preprint arXiv:1901.01536* (2019).

1367. Luong, N. C. *et al.* Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Commun. Surv. & Tutorials* **21**, 3133–3174 (2019).

1368. Di Felice, M., Bedogni, L. & Bononi, L. *Reinforcement Learning-Based Spectrum Management for Cognitive Radio Networks: A Literature Review and Case Study*, 1–38 (Springer Singapore, Singapore, 2018).

1369. Han, M. *et al.* A Review of Reinforcement Learning Methodologies for Controlling Occupant Comfort in Buildings. *Sustain. Cities Soc.* **51**, 101748 (2019).

1370. Mason, K. & Grijalva, S. A Review of Reinforcement Learning for Autonomous Building Energy Management. *Comput. & Electr. Eng.* **78**, 300–312 (2019).

1371. Mnih, V. *et al.* Human-Level Control Through Deep Reinforcement Learning. *Nature* **518**, 529–533 (2015).

1372. Nguyen, H. & La, H. Review of Deep Reinforcement Learning for Robot Manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 590–595 (IEEE, 2019).

1373. Bhagat, S., Banerjee, H., Ho Tse, Z. T. & Ren, H. Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges. *Robotics* **8**, 4 (2019).

1374. Zhao, T., Hachiya, H., Niu, G. & Sugiyama, M. Analysis and Improvement of Policy Gradient Estimation. In *Advances in Neural Information Processing Systems*, 262–270 (2011).

1375. Weng, L. Exploration strategies in deep reinforcement learning. Online: https://lilianweng.github.io/lil-log/2020/06/07/exploration-strategies-in-deep-reinforcement-learning.html (2020).

1376. Plappert, M. *et al.* Parameter Space Noise for Exploration. *arXiv preprint arXiv:1706.01905* (2018).

1377. Uhlenbeck, G. E. & Ornstein, L. S. On the Theory of the Brownian Motion. *Phys. Rev.* **36**, 823 (1930).

1378. Fujimoto, S., Van Hoof, H. & Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv preprint arXiv:1802.09477* (2018).

1379. Barth-Maron, G. *et al.* Distributed Distributional Deterministic Policy Gradients. *arXiv preprint arXiv:1804.08617* (2018).

1380. Kosaka, N. *Has it Explored Enough?* Master's thesis, Royal Holloway University of London, DOI: 10.13140/RG.2.2.11584.89604 (2019).

1381. Fortunato, M. *et al.* Noisy Networks for Exploration. *arXiv preprint arXiv:1706.10295* (2019).

1382. Hazan, E., Kakade, S., Singh, K. & Van Soest, A. Provably Efficient Maximum Entropy Exploration. In *International Conference on Machine Learning*, 2681–2691 (2019).

1383. Haarnoja, T., Tang, H., Abbeel, P. & Levine, S. Reinforcement Learning with Deep Energy-Based Policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1352–1361 (2017).

1384. Ahmed, Z., Le Roux, N., Norouzi, M. & Schuurmans, D. Understanding the Impact of Entropy on Policy Optimization. In *International Conference on Machine Learning*, 151–160 (2019).

1385. Aubret, A., Matignon, L. & Hassas, S. A Survey on Intrinsic Motivation in Reinforcement Learning. *arXiv preprint arXiv:1908.06976* (2019).

1386. Linke, C., Ady, N. M., White, M., Degris, T. & White, A. Adapting Behaviour via Intrinsic Reward: A Survey and Empirical Study. *arXiv preprint arXiv:1906.07865* (2019).

1387. Pathak, D., Agrawal, P., Efros, A. A. & Darrell, T. Curiosity-Driven Exploration by Self-Supervised Prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 16–17 (2017).

1388. Hoi, S. C., Sahoo, D., Lu, J. & Zhao, P. Online Learning: A Comprehensive Survey. *arXiv preprint arXiv:1802.02871* (2018).

1389. Wei, C.-Y., Hong, Y.-T. & Lu, C.-J. Online Reinforcement Learning in Stochastic Games. In *Advances in Neural Information Processing Systems*, 4987–4997 (2017).

1390. Levine, S., Kumar, A., Tucker, G. & Fu, J. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643* (2020).

1391. Seita, D. Offline (Batch) Reinforcement Learning: A Review of Literature and Applications. Seita's Place, Online: https://danieltakeshi.github.io/2020/06/28/offline-rl (2020).

1392. Fedus, W. *et al.* Revisiting Fundamentals of Experience Replay. *arXiv preprint arXiv:2007.06700* (2020).

1393. Nair, A., Dalal, M., Gupta, A. & Levine, S. Accelerating Online Reinforcement Learning with Offline Datasets. *arXiv preprint arXiv:2006.09359* (2020).

1394. Lin, L.-J. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Mach. Learn.* **8**, 293–321 (1992).

1395. Zhang, S. & Sutton, R. S. A Deeper Look at Experience Replay. *arXiv preprint arXiv:1712.01275* (2017).

1396. He, X., Zhao, K. & Chu, X. AutoML: A Survey of the State-of-the-Art. *arXiv preprint arXiv:1908.00709* (2019).

1397. Malekhosseini, E., Hajabdollahi, M., Karimi, N. & Samavi, S. Modeling Neural Architecture Search Methods for Deep Networks. *arXiv preprint arXiv:1912.13183* (2019).

1398. Jaafra, Y., Laurent, J. L., Deruyver, A. & Naceur, M. S. Reinforcement Learning for Neural Architecture Search: A Review. *Image Vis. Comput.* **89**, 57–66 (2019).

1399. Elsken, T., Metzen, J. H. & Hutter, F. Neural Architecture Search: A Survey. *arXiv preprint arXiv:1808.05377* (2018).

1400. Weill, C. *et al.* AdaNet: A Scalable and Flexible Framework for Automatically Learning Ensembles (2019). 1905.00080.

1401. Weill, C. Introducing AdaNet: Fast and Flexible AutoML with Learning Guarantees. Google AI Blog, Online: https://ai.googleblog.com/2018/10/introducing-adanet-fast-and-flexible.html (2018).

1402. Liu, C. *et al.* Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 82–92 (2019).

1403. Gong, X., Chang, S., Jiang, Y. & Wang, Z. AutoGAN: Neural Architecture Search for Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 3224–3234 (2019).

1404. Jin, H., Song, Q. & Hu, X. Auto-Keras: An Efficient Neural Architecture Search System. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1946–1956 (2019).

1405. Feurer, M. *et al.* Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*, 2962–2970 (2015).

1406. Liang, H. *et al.* DARTS+: Improved Differentiable Architecture Search with Early Stopping. *arXiv preprint arXiv:1909.06035* (2019).

1407. Molino, P., Dudin, Y. & Miryala, S. S. Ludwig: A Type-Based Declarative Deep Learning Toolbox. *arXiv preprint arXiv:1909.07930* (2019).

1408. Young, S. R., Rose, D. C., Karnowski, T. P., Lim, S.-H. & Patton, R. M. Optimizing Deep Learning Hyper-Parameters Through an Evolutionary Algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, 1–5 (2015).

1409. Patton, R. M. *et al.* 167-PFLOPS Deep Learning for Electron Microscopy: From Learning Physics to Atomic Manipulation. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, 638–648 (IEEE, 2018).

1410. Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B. & Xing, E. P. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. In *Advances in Neural Information Processing Systems*, 2016–2025 (2018).

1411. Nayman, N. *et al.* XNAS: Neural Architecture Search with Expert Advice. In *Advances in Neural Information Processing Systems*, 1977–1987 (2019).

1412. Jiang, W. *et al.* Accuracy vs. Efficiency: Achieving Both Through FPGA-Implementation Aware Neural Architecture Search. In *Proceedings of the 56th Annual Design Automation Conference 2019*, 1–6 (2019).

1413. Liu, C. *et al.* Progressive Neural Architecture Search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 19–34 (2018).

1414. Zhang, C., Ren, M. & Urtasun, R. Graph Hypernetworks for Neural Architecture Search. *arXiv preprint arXiv:1810.05749* (2018).

1415. Baker, B., Gupta, O., Raskar, R. & Naik, N. Accelerating Neural Architecture Search Using Performance Prediction. *arXiv preprint arXiv:1705.10823* (2017).

1416. Zoph, B. & Le, Q. V. Neural Architecture Search with Reinforcement Learning. *arXiv preprint arXiv:1611.01578* (2016).

1417. Godoy, D. Hyper-Parameters in Action! Part II – Weight Initializers. Towards Data Science, Online: https://towardsdatascience.com/hyper-parameters-in-action-part-ii-weight-initializers-35aee1a28404 (2018).

1418. Nagarajan, V. & Kolter, J. Z. Generalization in Deep Networks: The Role of Distance From Initialization. *arXiv preprint arXiv:1901.01672* (2019).

1419. Glorot, X. & Bengio, Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256 (2010).

1420. Kumar, S. K. On Weight Initialization in Deep Neural Networks. *arXiv preprint arXiv:1704.08863* (2017).

1421. Saxe, A. M., McClelland, J. L. & Ganguli, S. Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks. *arXiv preprint arXiv:1312.6120* (2013).

1422. Henaff, M., Szlam, A. & LeCun, Y. Recurrent Orthogonal Networks and Long-Memory Tasks. *arXiv preprint arXiv:1602.06662* (2016).

1423. Le, Q. V., Jaitly, N. & Hinton, G. E. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv preprint arXiv:1504.00941* (2015).

1424. Mikolov, T., Joulin, A., Chopra, S., Mathieu, M. & Ranzato, M. Learning Longer Memory in Recurrent Neural Networks. *arXiv preprint arXiv:1412.7753* (2014).

1425. Pitis, S. Non-Zero Initial States for Recurrent Neural Networks. Online: https://r2rt.com/non-zero-initial-states-for-recurrent-neural-networks.html (2016).

1426. Mishkin, D. & Matas, J. All You Need is a Good Init. *arXiv preprint arXiv:1511.06422* (2015).

1427. Sussillo, D. & Abbott, L. Random Walk Initialization for Training Very Deep Feedforward Networks. *arXiv preprint arXiv:1412.6558* (2014).

1428. Dauphin, Y. N. & Schoenholz, S. MetaInit: Initializing Learning by Learning to Initialize. In *Advances in Neural Information Processing Systems*, 12645–12657 (2019).

1429. Kukačka, J., Golkov, V. & Cremers, D. Regularization for Deep Learning: A Taxonomy. *arXiv preprint arXiv:1710.10686* (2017).

1430. Kang, G. *Regularization in Deep Neural Networks*. Ph.D. thesis, University of Technology Sydney (2019).

1431. Liu, Z., Li, X., Kang, B. & Darrell, T. Regularization Matters in Policy Optimization. *arXiv preprint arXiv:1910.09191* (2019).

1432. Vettam, S. & John, M. Regularized Deep Learning with Non-Convex Penalties. *arXiv preprint arXiv:1909.05142* (2019).

1433. Golatkar, A. S., Achille, A. & Soatto, S. Time Matters in Regularizing Deep Networks: Weight Decay and Data Augmentation Affect Early Learning Dynamics, Matter Little Near Convergence. In *Advances in Neural Information Processing Systems*, 10678–10688 (2019).

1434. Tanay, T. & Griffin, L. D. A New Angle on L2 Regularization. *arXiv preprint arXiv:1806.11186* (2018).

1435. Van Laarhoven, T. L2 Regularization versus Batch and Weight Normalization. *arXiv preprint arXiv:1706.05350* (2017).

1436. Van Den Doel, K., Ascher, U. & Haber, E. The Lost Honour of L2-Based Regularization. *Large Scale Inverse Probl.* **13**, 181–203 (2012).

1437. Gribonval, R., Cevher, V. & Davies, M. E. Compressible Distributions for High-Dimensional Statistics. *IEEE Transactions on Inf. Theory* **58**, 5016–5034 (2012).

1438. Ng, A. Y. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 78 (2004).

1439. Zou, H. & Hastie, T. Regularization and Variable Selection via the Elastic Net. *J. Royal Stat. Soc. Ser. B (Statistical Methodol.* **67**, 301–320 (2005).

1440. Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *J. Royal Stat. Soc. Ser. B (Methodological)* **58**, 267–288 (1996).

1441. Hoerl, A. E. & Kennard, R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **12**, 55–67 (1970).

1442. Zhang, J., He, T., Sra, S. & Jadbabaie, A. Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. *arXiv preprint arXiv:1905.11881* (2019).

1443. Chen, X., Wu, Z. S. & Hong, M. Understanding Gradient Clipping in Private SGD: A Geometric Perspective. *arXiv preprint arXiv:2006.15429* (2020).

1444. Menon, A. K., Rawat, A. S., Reddi, S. J. & Kumar, S. Can Gradient Clipping Mitigate Label Noise? In *International Conference on Learning Representations* (2019).

1445. Bengio, Y., Boulanger-Lewandowski, N. & Pascanu, R. Advances in Optimizing Recurrent Networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8624–8628 (IEEE, 2013).

1446. Chen, M. X. *et al.* The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. *arXiv preprint arXiv:1804.09849* (2018).

1447. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The J. Mach. Learn. Res.* **15**, 1929–1958 (2014).

1448. Labach, A., Salehinejad, H. & Valaee, S. Survey of Dropout Methods for Deep Neural Networks. *arXiv preprint arXiv:1904.13310* (2019).

1449. Li, Z., Gong, B. & Yang, T. Improved Dropout for Shallow and Deep Learning. In *Advances in Neural Information Processing Systems*, 2523–2531 (2016).

1450. Mianjy, P., Arora, R. & Vidal, R. On the Implicit Bias of Dropout. In *International Conference on Machine Learning*, 3540–3548 (2018).

1451. Warde-Farley, D., Goodfellow, I. J., Courville, A. & Bengio, Y. An Empirical Analysis of Dropout in Piecewise Linear Networks. *arXiv preprint arXiv:1312.6197* (2013).

1452. Garbin, C., Zhu, X. & Marques, O. Dropout vs. Batch Normalization: An Empirical Study of Their Impact to Deep Learning. *Multimed. Tools Appl.* 1–39 (2020).

1453. Cai, S. *et al.* Effective and Efficient Dropout for Deep Convolutional Neural Networks. *arXiv preprint arXiv:1904.03392* (2019).

1454. Ghiasi, G., Lin, T.-Y. & Le, Q. V. DropBlock: A Regularization Method for Convolutional Networks. In *Advances in Neural Information Processing Systems*, 10727–10737 (2018).

1455. Faramarzi, M., Amini, M., Badrinaaraayanan, A., Verma, V. & Chandar, S. PatchUp: A Regularization Technique for Convolutional Neural Networks. *arXiv preprint arXiv:2006.07794* (2020).

1456. Kang, G., Li, J. & Tao, D. Shakeout: A New Approach to Regularized Deep Neural Network Training. *IEEE Transactions on Pattern Analysis Mach. Intell.* **40**, 1245–1258 (2017).

1457. Kang, G., Li, J. & Tao, D. Shakeout: A New Regularized Deep Neural Network Training Scheme. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 1751–1757 (2016).

1458. Zhou, M. *et al.* Towards Understanding the Importance of Noise in Training Neural Networks. *arXiv preprint arXiv:1909.03172* (2019).

1459. Graves, A., Mohamed, A.-r. & Hinton, G. Speech Recognition with Deep Recurrent Neural Networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6645–6649 (IEEE, 2013).

1460. Graves, A. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems*, 2348–2356 (2011).

1461. Sum, J., Leung, C.-S. & Ho, K. A Limitation of Gradient Descent Learning. *IEEE Transactions on Neural Networks Learn. Syst.* (2019).

1462. Holmstrom, L. & Koistinen, P. Using Additive Noise in Back-Propagation Training. *IEEE Transactions on Neural Networks* **3**, 24–38 (1992).

1463. You, Z., Ye, J., Li, K., Xu, Z. & Wang, P. Adversarial Noise Layer: Regularize Neural Network by Adding Noise. In *2019 IEEE International Conference on Image Processing (ICIP)*, 909–913 (IEEE, 2019).

1464. Jenni, S. & Favaro, P. On Stabilizing Generative Adversarial Training with Noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12145–12153 (2019).

1465. Sun, Y., Tian, Y., Xu, Y. & Li, J. Limited Gradient Descent: Learning With Noisy Labels. *IEEE Access* **7**, 168296–168306 (2019).

1466. Simsekli, U., Sagun, L. & Gurbuzbalaban, M. A Tail-Index Analysis of Stochastic Gradient Noise in Deep Neural Networks. *arXiv preprint arXiv:1901.06053* (2019).

1467. Neelakantan, A. *et al.* Adding Gradient Noise Improves Learning for Very Deep Networks. *arXiv preprint arXiv:1511.06807* (2015).

1468. Shorten, C. & Khoshgoftaar, T. M. A Survey on Image Data Augmentation for Deep Learning. *J. Big Data* **6**, 60 (2019).

1469. Raileanu, R., Goldstein, M., Yarats, D., Kostrikov, I. & Fergus, R. Automatic Data Augmentation for Generalization in Deep Reinforcement Learning. *arXiv preprint arXiv:2006.12862* (2020).

1470. Antczak, K. On Regularization Properties of Artificial Datasets for Deep Learning. *arXiv preprint arXiv:1908.07005* (2019).

1471. Ouali, Y., Hudelot, C. & Tami, M. An Overview of Deep Semi-Supervised Learning. *arXiv preprint arXiv:2006.05278* (2020).

1472. Zhu, J. Semi-Supervised Learning: the Case When Unlabeled Data is Equally Useful. *arXiv preprint arXiv:2005.11018* (2020).

1473. Aitchison, L. A Statistical Theory of Semi-Supervised Learning. *arXiv preprint arXiv:2008.05913* (2020).

1474. Bagherzadeh, J. & Asil, H. A Review of Various Semi-Supervised Learning Models with a Deep Learning and Memory Approach. *Iran J. Comput. Sci.* **2**, 65–80 (2019).

1475. Rasmus, A., Berglund, M., Honkala, M., Valpola, H. & Raiko, T. Semi-Supervised Learning with Ladder Networks. In *Advances in Neural Information Processing Systems*, 3546–3554 (2015).

1476. Lee, D.-H. Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. In *Workshop on Challenges in Representation Learning, ICML*, vol. 3 (2013).

1477. Sun, S., Mao, L., Dong, Z. & Wu, L. Multiview Transfer Learning and Multitask Learning. In *Multiview Machine Learning*, 85–104 (Springer, 2019).

1478. Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098* (2017).

1479. Thung, K.-H. & Wee, C.-Y. A Brief Review on Multi-Task Learning. *Multimed. Tools Appl.* **77**, 29705–29725 (2018).

1480. Zhang, Y. & Yang, Q. A Survey on Multi-Task Learning. *arXiv preprint arXiv:1707.08114* (2017).

1481. Caruana, R. Multitask Learning. *Mach. Learn.* **28**, 41–75 (1997).

1482. Odena, A., Olah, C. & Shlens, J. Conditional Image Synthesis With Auxiliary Classifier GANs. *arXiv preprint arXiv:1610.09585* (2016).

1483. Shu, R., Bui, H. & Ermon, S. AC-GAN Learns a Biased Distribution. In *NIPS Workshop on Bayesian Deep Learning*, vol. 8 (2017).

1484. Gong, M., Xu, Y., Li, C., Zhang, K. & Batmanghelich, K. Twin Auxilary Classifiers GAN. In *Advances in Neural Information Processing Systems*, 1330–1339 (2019).

1485. Han, L., Stathopoulos, A., Xue, T. & Metaxas, D. Unbiased Auxiliary Classifier GANs with MINE. *arXiv preprint arXiv:2006.07567* (2020).

1486. Better Performance with the tf.data API. TensorFlow Documentation, Online: https://www.tensorflow.org/guide/data_performance (2020).

1487. Li, B., Wu, F., Lim, S.-N., Belongie, S. & Weinberger, K. Q. On feature normalization and data augmentation. *arXiv preprint arXiv:2002.11102* (2020).

1488. Bhanja, S. & Das, A. Impact of Data Normalization on Deep Neural Network for Time Series Forecasting. *arXiv preprint arXiv:1812.05519* (2018).

1489. van Hasselt, H. P., Guez, A., Hessel, M., Mnih, V. & Silver, D. Learning Values Across Many Orders of Magnitude. In *Advances in Neural Information Processing Systems*, 4287–4295 (2016).

1490. Li, M., Soltanolkotabi, M. & Oymak, S. Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 4313–4324 (2020).

1491. Flynn, T., Yu, K. M., Malik, A., D'Imperio, N. & Yoo, S. Bounding the Expected Run-Time of Nonconvex Optimization with Early Stopping. *arXiv preprint arXiv:2002.08856* (2020).

1492. Nagaraj, D., Jain, P. & Netrapalli, P. SGD Without Replacement: Sharper Rates for General Smooth Convex Functions. In *International Conference on Machine Learning*, 4703–4711 (2019).

1493. Gürbüzbalaban, M., Ozdaglar, A. & Parrilo, P. Why Random Reshuffling Beats Stochastic Gradient Descent. *Math. Program.* 1–36 (2019).

1494. Haochen, J. & Sra, S. Random Shuffling Beats SGD After Finite Epochs. In *International Conference on Machine Learning*, 2624–2633 (2019).

1495. Shamir, O. Without-Replacement Sampling for Stochastic Gradient Methods. In *Advances in Neural Information Processing Systems*, 46–54 (2016).

1496. Bottou, L. Curiously Fast Convergence of Some Stochastic Gradient Descent Algorithms. In *Proceedings of the Symposium on Learning and Data Science* (2009).

1497. tf.data.Dataset. TensorFlow Documentation, Online: https://www.tensorflow.org/api_docs/python/tf/data/Dataset (2020).

1498. Harrington, P. d. B. Multiple Versus Single Set Validation of Multivariate Models to Avoid Mistakes. *Critical Rev. Anal. Chem.* **48**, 33–46 (2018).

1499. Breiman, L. Bagging Predictors. *Mach. Learn.* **24**, 123–140 (1996).

1500. Breiman, L. Random Forests. *Mach. Learn.* **45**, 5–32 (2001).

1501. Goel, E., Abhilasha, E., Goel, E. & Abhilasha, E. Random Forest: A Review. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **7**, 251–257 (2017).

1502. Probst, P., Wright, M. N. & Boulesteix, A.-L. Hyperparameters and Tuning Strategies for Random Forest. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **9**, e1301 (2019).

1503. Xu, Y. & Goodacre, R. On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. *J. Analysis Test.* **2**, 249–262 (2018).

1504. Guyon, I. A Scaling Law for the Validation-Set Training-Set Size Ratio. *AT&T Bell Lab.* **1** (1997).

1505. Newman, M. E. J. Power Laws, Pareto Distributions and Zipf's Law. *Contemp. Phys.* **46**, 323–351 (2005).

1506. Opeyemi, B. Deployment of Machine Learning Models Demystified (Part 1). Towards Data Science, Online: https://towardsdatascience.com/deployment-of-machine-learning-model-demystified-part-1-1181d91815d2 (2019).

1507. Opeyemi, B. Deployment of Machine Learning Model Demystified (Part 2). Medium, Online: https://medium.com/@opeyemibami/deployment-of-machine-learning-models-demystified-part-2-63eadaca1571 (2019).

1508. Wu, C.-J. *et al.* Machine Learning at Facebook: Understanding Inference at the Edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 331–344 (IEEE, 2019).

1509. Cai, H., Gan, C. & Han, S. Once for All: Train One Network and Specialize it for Efficient Deployment. *arXiv preprint arXiv:1908.09791* (2019).

1510. Suresh, A. & Ganesh Kumar, P. Optimization of Metascheduler for Cloud Machine Learning Services. *Wirel. Pers. Commun.* 1–22 (2020).

1511. Kumar, Y., Kaul, S. & Sood, K. Effective Use of the Machine Learning Approaches on Different Clouds. In *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India* (2019).

1512. Dubois, D. J., Trubiani, C. & Casale, G. Model-driven Application Refactoring to Minimize Deployment Costs in Preemptible Cloud Resources. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 335–342 (IEEE, 2016).

1513. Oracle *et al.* GraphPipe: Machine Learning Model Deployment Made Simple.

1514. FlatBuffers: Memory Efficient Serialization Library. FlatBuffers Documentation, Online: https://google.github.io/flatbuffers (2020).

1515. Blalock, D., Ortiz, J. J. G., Frankle, J. & Guttag, J. What is the State of Neural Network Pruning? *arXiv preprint arXiv:2003.03033* (2020).

1516. Pasandi, M. M., Hajabdollahi, M., Karimi, N. & Samavi, S. Modeling of Pruning Techniques for Deep Neural Networks Simplification. *arXiv preprint arXiv:2001.04062* (2020).

1517. Wu, H., Judd, P., Zhang, X., Isaev, M. & Micikevicius, P. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. *arXiv preprint arXiv:2004.09602* (2020).

1518. Nayak, P., Zhang, D. & Chai, S. Bit Efficient Quantization for Deep Neural Networks. *arXiv preprint arXiv:1910.04877* (2019).

1519. Zhou, Y., Moosavi-Dezfooli, S.-M., Cheung, N.-M. & Frossard, P. Adaptive Quantization for Deep Neural Network. *arXiv preprint arXiv:1712.01048* (2017).

1520. Yang, J. *et al.* Quantization Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7308–7316 (2019).

1521. Zhuang, B. *et al.* Effective Training of Convolutional Neural Networks with Low-Bitwidth Weights and Activations. *arXiv preprint arXiv:1908.04680* (2019).

1522. Li, H. *et al.* Training Quantized Nets: A Deeper Understanding. In *Advances in Neural Information Processing Systems*, 5811–5821 (2017).

1523. Wang, S. & Kanwar, P. BFloat16: The Secret to High Performance on Cloud TPUs. Google Cloud, Online: https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus (2019).

1524. Marco, V. S., Taylor, B., Wang, Z. & Elkhatib, Y. Optimizing Deep Learning Inference on Embedded Systems Through Adaptive Model Selection. *ACM Transactions on Embed. Comput. Syst. (TECS)* **19**, 1–28 (2020).

1525. Jackson, B. How to Optimize Images for Web and Performance. Kinsta Blog, Online: https://kinsta.com/blog/optimize-images-for-web (2020).

1526. Leventić, H., Nenadić, K., Galić, I. & Livada, Č. Compression Parameters Tuning for Automatic Image Optimization in Web Applications. In *2016 International Symposium ELMAR*, 181–184 (IEEE, 2016).

1527. Olah, C., Mordvintsev, A. & Schubert, L. Feature Visualization. Distill, Online: https://distill.pub/2017/feature-visualization (2017).

1528. Xie, N., Ras, G., van Gerven, M. & Doran, D. Explainable Deep Learning: A Field Guide for the Uninitiated. *arXiv preprint arXiv:2004.14545* (2020).

1529. Vilone, G. & Longo, L. Explainable Artificial Intelligence: A Systematic Review. *arXiv preprint arXiv:2006.00093* (2020).

1530. Arrieta, A. B. *et al.* Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI. *Inf. Fusion* **58**, 82–115 (2020).

1531. Puiutta, E. & Veith, E. Explainable Reinforcement Learning: A Survey. *arXiv preprint arXiv:2005.06247* (2020).

1532. Gunning, D. & Aha, D. W. DARPA's Explainable Artificial Intelligence Program. *AI Mag.* **40**, 44–58 (2019).

1533. Samek, W. & Müller, K.-R. Towards Explainable Artificial Intelligence. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 5–22 (Springer, 2019).

1534. Hase, P. & Bansal, M. Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior? *arXiv preprint arXiv:2005.01831* (2020).

1535. Rebuffi, S.-A., Fong, R., Ji, X. & Vedaldi, A. There and Back Again: Revisiting Backpropagation Saliency Methods. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8839–8848 (2020).

1536. Wang, Y., Su, H., Zhang, B. & Hu, X. Learning Reliable Visual Saliency for Model Explanations. *IEEE Transactions on Multimed.* (2019).

1537. Kim, B. *et al.* Why are Saliency Maps Noisy? Cause of and Solution to Noisy Saliency Maps. *arXiv preprint arXiv:1902.04893* (2019).

1538. Selvaraju, R. R. *et al.* Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, 618–626 (2017).

1539. Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P. & Boracchi, G. Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4067–4071 (IEEE, 2020).

1540. Omeiza, D., Speakman, S., Cintas, C. & Weldermariam, K. Smooth Grad-CAM++: An Enhanced Inference Level Visualization Technique for Deep Convolutional Neural Network Models. *arXiv preprint arXiv:1908.01224* (2019).

1541. Chattopadhay, A., Sarkar, A., Howlader, P. & Balasubramanian, V. N. Grad-Cam++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 839–847 (IEEE, 2018).

1542. Patro, B. N., Lunayach, M., Patel, S. & Namboodiri, V. P. U-Cam: Visual Explanation Using Uncertainty Based Class Activation Maps. In *Proceedings of the IEEE International Conference on Computer Vision*, 7444–7453 (2019).

1543. Ullah, I. *et al.* A Brief Survey of Visual Saliency Detection. *Multimed. Tools Appl.* 1–41 (2020).

1544. Borji, A. Saliency Prediction in the Deep Learning Era: Successes and Limitations. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2019).

1545. Wang, W. *et al.* Revisiting Video Saliency Prediction in the Deep Learning Era. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2019).

1546. Chen, L., Chen, J., Hajimirsadeghi, H. & Mori, G. Adapting Grad-CAM for Embedding Networks. In *The IEEE Winter Conference on Applications of Computer Vision*, 2794–2803 (2020).

1547. Ramaswamy, H. G. *et al.* Ablation-CAM: Visual Explanations for Deep Convolutional Network via Gradient-free Localization. In *The IEEE Winter Conference on Applications of Computer Vision*, 983–991 (2020).

1548. Wang, H. *et al.* Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 24–25 (2020).

1549. Cancela, B., Bolón-Canedo, V., Alonso-Betanzos, A. & Gama, J. A Scalable Saliency-Based Feature Selection Method with Instance-Level Information. *Knowledge-Based Syst.* **192**, 105326 (2020).

1550. Nguyen, A., Yosinski, J. & Clune, J. Understanding Neural Networks via Feature Visualization: A Survey. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 55–76 (Springer, 2019).

1551. Xiao, W. & Kreiman, G. Gradient-Free Activation Maximization for Identifying Effective Stimuli. *arXiv preprint arXiv:1905.00378* (2019).

1552. Erhan, D., Bengio, Y., Courville, A. & Vincent, P. Visualizing Higher-Layer Features of a Deep Network. *Univ. Montr.* **1341** (2009).

1553. Mordvintsev, A., Olah, C. & Tyka, M. Inceptionism: Going Deeper into Neural Networks. Google AI Blog, Online: https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html (2015).

1554. Maaten, L. v. d. & Hinton, G. Visualizing Data Using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).

1555. Wattenberg, M., Viégas, F. & Johnson, I. How to Use t-SNE Effectively. *Distill* **1**, e2 (2016).

1556. Van Der Maaten, L. Barnes-Hut-SNE. *arXiv preprint arXiv:1301.3342* (2013).

1557. Barnes, J. & Hut, P. A Hierarchical$o(n \log n)$ Force-Calculation Algorithm. *Nature* **324**, 446–449 (1986).

1558. Wang, Z. J. *et al.* CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *arXiv preprint arXiv:2004.15004* (2020).

1559. Wang, Z. J. *et al.* CNN 101: Interactive Visual Learning for Convolutional Neural Networks. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–7 (2020).

1560. Kahng, M., Thorat, N., Chau, D. H. P., Viégas, F. B. & Wattenberg, M. GAN Lab: Understanding Complex Deep Generative Models Using Interactive Visual Experimentation. *IEEE Transactions on Vis. Comput. Graph.* **25**, 1–11 (2018).

1561. Gangavarapu, T., Jaidhar, C. & Chanduka, B. Applicability of Machine Learning in Spam and Phishing Email Filtering: Review and Approaches. *Artif. Intell. Rev.* 1–63 (2020).

1562. Dada, E. G. *et al.* Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems. *Heliyon* **5**, e01802 (2019).

1563. Bhuiyan, H., Ashiquzzaman, A., Juthi, T. I., Biswas, S. & Ara, J. A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques. *Glob. J. Comput. Sci. Technol.* (2018).

1564. Zhang, J. & Zeng, W. Mining Scientific and Technical Literature: From Knowledge Extraction to Summarization. In *Trends and Applications of Text Summarization Techniques* (IGI Global, 2020).

1565. Dangovski, R., Jing, L., Nakov, P., Tatalović, M. & Soljačić, M. Rotational Unit of Memory: A Novel Representation Unit for RNNs with Scalable Applications. *Transactions Assoc. for Comput. Linguist.* **7**, 121–138 (2019).

1566. Scholarcy: The AI-Powered Article Summarizer. Online: https://www.scholarcy.com (2020).

1567. Romanov, A., Lomotin, K. & Kozlova, E. Application of Natural Language Processing Algorithms to the Task of Automatic Classification of Russian Scientific Texts. *Data Sci. J.* **18** (2019).

1568. Gonçalves, S., Cortez, P. & Moro, S. A Deep Learning Classifier for Sentence Classification in Biomedical and Computer Science Abstracts. *Neural Comput. Appl.* 1–15 (2019).

1569. Hughes, M., Li, I., Kotoulas, S. & Suzumura, T. Medical Text Classification Using Convolutional Neural Networks. *Stud. Heal. Technol. Informatics* **235**, 246–50 (2017).

1570. Liu, J., Xu, Y. & Zhu, Y. Automated Essay Scoring Based on Two-Stage Learning. *arXiv preprint arXiv:1901.07744* (2019).

1571. Dong, F., Zhang, Y. & Yang, J. Attention-Based Recurrent Convolutional Neural Network for Automatic Essay Scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 153–162 (2017).

1572. Taghipour, K. & Ng, H. T. A Neural Approach to Automated Essay Scoring. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, 1882–1891 (2016).

1573. Alikaniotis, D., Yannakoudakis, H. & Rei, M. Automatic Text Scoring Using Neural Networks. *arXiv preprint arXiv:1606.04289* (2016).

1574. Foltỳnek, T., Meuschke, N. & Gipp, B. Academic Plagiarism Detection: A Systematic Literature Review. *ACM Comput. Surv. (CSUR)* **52**, 1–42 (2019).

1575. Meuschke, N., Stange, V., Schubotz, M., Kramer, M. & Gipp, B. Improving Academic Plagiarism Detection for STEM documents by Analyzing Mathematical Content and Citations. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 120–129 (IEEE, 2019).

1576. Ullah, F., Wang, J., Farhan, M., Habib, M. & Khalid, S. Software Plagiarism Detection in Multiprogramming Languages Using Machine Learning Approach. *Concurr. Comput. Pract. Exp.* e5000 (2018).

1577. Lakkaraju, H. *et al.* A Machine Learning Framework to Identify Students at Risk of Adverse Academic Outcomes. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1909–1918 (2015).

1578. Foster, D. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play* (O'Reilly Media, 2019).

1579. Zhan, H., Dai, L. & Huang, Z. Deep Learning in the Field of Art. In *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science*, 717–719 (2019).

1580. Gatys, L. A., Ecker, A. S. & Bethge, M. Image Style Transfer Using Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2414–2423 (2016).

1581. Gatys, L., Ecker, A. & Bethge, M. A Neural Algorithm of Artistic Style. *Nat. Commun.* (2015).

1582. Dhariwal, P. *et al.* Jukebox: A Generative Model for Music. *arXiv preprint arXiv:2005.00341* (2020).

1583. Briot, J.-P. & Pachet, F. Deep Learning for Music Generation: Challenges and Directions. *Neural Comput. Appl.* **32**, 981–993 (2020).

1584. Briot, J.-P., Hadjeres, G. & Pachet, F.-D. *Deep Learning Techniques for Music Generation* (Springer, 2020).

1585. Brown, T. B. *et al.* Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165* (2020).

1586. Radford, A. *et al.* Better Language Models and Their Implications. OpenAI Blog, Online: https://openai.com/blog/better-language-models (2019).

1587. Chen, H., Le, T. H. M. & Babar, M. A. Deep Learning for Source Code Modeling and Generation: Models, Applications and Challenges. *ACM Comput. Surv. (CSUR)* (2020).

1588. Allamanis, M., Barr, E. T., Devanbu, P. & Sutton, C. A Survey of Machine Learning for Big Code and Naturalness. *ACM Comput. Surv. (CSUR)* **51**, 1–37 (2018).

1589. Autocompletion with deep learning. TabNine Blog, Online: https://www.tabnine.com/blog/deep (2019).

1590. Svyatkovskiy, A., Deng, S. K., Fu, S. & Sundaresan, N. IntelliCode Compose: Code Generation Using Transformer. *arXiv preprint arXiv:2005.08025* (2020).

1591. Hammad, M., Babur, Ö., Basit, H. A. & Brand, M. v. d. DeepClone: Modeling Clones to Generate Code Predictions. *arXiv preprint arXiv:2007.11671* (2020).

1592. Schuster, R., Song, C., Tromer, E. & Shmatikov, V. You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion. *arXiv preprint arXiv:2007.02220* (2020).

1593. Svyatkovskoy, A. *et al.* Fast and Memory-Efficient Neural Code Completion. *arXiv preprint arXiv:2004.13651* (2020).

1594. Hellendoorn, V. J., Proksch, S., Gall, H. C. & Bacchelli, A. When Code Completion Fails: A Case Study on Real-World Completions. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 960–970 (IEEE, 2019).

1595. Balog, M., Gaunt, A. L., Brockschmidt, M., Nowozin, S. & Tarlow, D. DeepCoder: Learning to Write Programs. In *International Conference on Learning Representations (ICLR 2017)* (OpenReview.net, 2017).

1596. Murali, V., Qi, L., Chaudhuri, S. & Jermaine, C. Neural Sketch Learning for Conditional Program Generation. *arXiv preprint arXiv:1703.05698* (2018).

1597. Demir, S., Mutlu, U. & Özdemir, Ö. Neural Academic Paper Generation. *arXiv preprint arXiv:1912.01982* (2019).

1598. SciNote. Manuscript Writer. Online: https://www.scinote.net/manuscript-writer (2020).

1599. Stribling, J., Krohn, M. & Aguayo, D. SCIgen - An Automatic CS Paper Generator. Online: https://pdos.csail.mit.edu/archive/scigen (2005).

1600. Raghu, M. & Schmidt, E. A Survey of Deep Learning for Scientific Discovery. *arXiv preprint arXiv:2003.11755* (2020).

1601. Kepner, J., Cho, K. & Claffy, K. New Phenomena in Large-Scale Internet Traffic. *arXiv preprint cs.NI/1904.04396* (2019).

1602. Adekitan, A. I., Abolade, J. & Shobayo, O. Data Mining Approach for Predicting the Daily Internet Data Traffic of a Smart University. *J. Big Data* **6**, 11 (2019).

1603. Xu, X., Wang, J., Peng, H. & Wu, R. Prediction of Academic Performance Associated with Internet Usage Behaviors Using Machine Learning Algorithms. *Comput. Hum. Behav.* **98**, 166–173 (2019).

1604. Granger, R. Toward the Quantification of Cognition. *arXiv preprint arXiv:2008.05580* (2020).

1605. Musk, E. *et al.* An Integrated Brain-Machine Interface Platform with Thousands of Channels. *J. Med. Internet Res.* **21**, e16194 (2019).

1606. Tshitoyan, V. *et al.* Unsupervised Word Embeddings Capture Latent Knowledge from Materials Science Literature. *Nature* **571**, 95–98 (2019).

1607. Ruf, J. & Wang, W. Neural Networks for Option Pricing and Hedging: A Literature Review. *J. Comput. Finance, Forthcom.* (2020).

1608. Huang, B., Huan, Y., Xu, L. D., Zheng, L. & Zou, Z. Automated Trading Systems Statistical and Machine Learning Methods and Hardware Implementation: A Survey. *Enterp. Inf. Syst.* **13**, 132–144 (2019).

1609. Raghavan, M., Barocas, S., Kleinberg, J. & Levy, K. Mitigating Bias in Algorithmic Hiring: Evaluating Claims and Practices. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 469–481 (2020).

1610. Mahmoud, A. A., Shawabkeh, T. A., Salameh, W. A. & Al Amro, I. Performance Predicting in Hiring Process and Performance Appraisals Using Machine Learning. In *2019 10th International Conference on Information and Communication Systems (ICICS)*, 110–115 (IEEE, 2019).

1611. Raub, M. Bots, Bias and Big Data: Artificial Intelligence, Algorithmic Bias and Disparate Impact Liability in Hiring Practices. *Arkansas Law Rev.* **71**, 529 (2018).

1612. Newman, N. Reengineering Workplace Bargaining: How Big Data Drives Lower Wages and How Reframing Labor Law can Restore Information Equality in the Workplace. *Univ. Cincinnati Law Rev.* **85**, 693 (2017).

1613. Price, W. & Nicholson, I. Grants. *Berkeley Technol. Law J.* **34**, 1 (2019).

1614. Zhuang, H. & Acuna, D. E. The Effect of Novelty on the Future Impact of Scientific Grants. *arXiv preprint arXiv:1911.02712* (2019).

1615. Zhang, W. E., Sheng, Q. Z., Alhazmi, A. & Li, C. Adversarial Attacks on Deep-Learning Models in Natural Language Processing: A survey. *ACM Transactions on Intell. Syst. Technol. (TIST)* **11**, 1–41 (2020).

1616. Ma, X. *et al.* Understanding Adversarial Attacks on Deep Learning Based Medical Image Analysis Systems. *Pattern Recognit.* 107332 (2020).

1617. Yuan, X., He, P., Zhu, Q. & Li, X. Adversarial examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks Learn. Syst.* **30**, 2805–2824 (2019).

1618. Akhtar, N. & Mian, A. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* **6**, 14410–14430 (2018).

1619. Goodfellow, I. J., Shlens, J. & Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572* (2014).

1620. Wen, Y., Li, S. & Jia, K. Towards Understanding the Regularization of Adversarial Robustness on Neural Networks. *OpenReview.net* (2019).

1621. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D. & Jana, S. Certified Robustness to Adversarial Examples with Differential Privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, 656–672 (IEEE, 2019).

1622. Li, Y. *et al.* Optimal Transport Classifier: Defending Against Adversarial Attacks by Regularized Deep Embedding. *arXiv preprint arXiv:1811.07950* (2018).

1623. Deniz, O., Pedraza, A., Vallez, N., Salido, J. & Bueno, G. Robustness to Adversarial Examples can be Improved With Overfitting. *Int. J. Mach. Learn. Cybern.* 1–10 (2020).

1624. Kinoshita, Y. & Kiya, H. Fixed Smooth Convolutional Layer for Avoiding Checkerboard Artifacts in CNNs. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3712–3716 (IEEE, 2020).

1625. Xiao, H., Rasul, K. & Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747* (2017).